RITU DHAWAN
Evaluation of Web Personalization Software and Visualization With the Help of Usability Study
(Under the Direction of EILEEN T. KRAEMER)

Evaluation is an important element in the creation of complex software. In this thesis, we describe two such evaluations. In the first study, we perform usability testing on web-based personalization software developed by John O' Looney of the Carl Vinson Institute of Government at the University of Georgia. Personalization of a web page involves dynamically altering the contents of the web pages according to the preferences of customers or clients, so that each client gets information specific to his needs. This study assists in and suggests ways to improve the usability and learnability of this software. The second study was a user study to determine if visualization is beneficial in helping users to learn about and understand distributed algorithms.

INDEX WORDS: Usability, Personalization, Visualization, Algorithms, Learnability

EVALUATION OF WEB PERSONALIZATION SOFTWARE AND VISUALIZATION WITH THE HELP OF USABILITY STUDY

by

RITU DHAWAN

B.S., Shahu Ji Maharaj Kanpur University, India, 1995

B.Ed., Shahu Ji Maharaj Kanpur University, India, 1996

MS, The University of Georgia 1999

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTERS OF SCIENCE

ATHNES, GEORGIA

2001

EVALUATION OF WEB PERSONALIZATION SOFTWARE AND VISUALIZATION WITH THE HELP OF USABILITY STUDY

by

RITU DHAWAN

Approved:

Major Professor: Eileen Kraemer

Committee: Hamid Arabnia

Daniel Everett

Electronic Version Approved:

Gordhan L. Patel
Dean of the Graduate School
The University of Georgia
December 20001

This thesis is dedicated to my parents and my loving husband

ACKNOWLEDGEMENTS

I wish to thank and express my sincere gratitude to my Major Advisor Dr. Eileen Kraemer for her constant support and encouragement to make this thesis successful. I would also like to thank my committee members Prof. Hamid Arabnia and Prof. Daniel Everett for their time and consideration. My thanks are also due to my project mates Rong Wu and Mihail Tudoreanu.

The author would also like to thank to the Department of Computer Science and the University of Georgia for providing the facilities to support this research. My special thanks to Dr. John O' Looney for letting me do the usability testing on his software.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Usability is an important feature of any software designed for direct use by humans. Research in evaluation techniques and methodologies has focused on helping designers to evaluate prototypes and software in order to identify problems and to promote ease-of-use, learnability and productivity. In this, thesis, we apply these techniques and methodologies to two types of software: a web personalization package, and program visualization software.

The web personalization package, developed by John O' Looney of the Carl Vinson Institute of Government at the University of Georgia, was designed for public managers to create personalized web pages for their clients. This software will help managers to generate web forms, which can be filled in by their clients to get information specific to them. Due to the emergence of low cost computer-mediated communications via the Internet and increases in online privacy protection, this technology is believed likely to be adopted by the public sector organizations in order to increase the efficiency and effectiveness of their work. This will also help the clients get important information from any place without personally visiting or talking to the manager. This technique is a low-overhead approach to reach millions of customers without spending excessive time and money (O'Leary, 1999). In spite of having these many capabilities, some barriers to the use of this software still exist. Managers in the public sector are less likely to have received training and education in the use of this technology and, as this software involves extensive use of computers, it is very important that it is easy for this target group to use. Another restriction is that investment in technology training is relatively low in the public sectors and thus it useful to have the software be in a form that is ready to use “out of the box”, without any professional training. Therefore, it is important to perform a usability study to determine if the product is in a form that promotes large scale acceptance of the software. The main goal of this usability study is to make this personalization software easy to use and learn.

Visualization software programs use graphics and animation techniques to show program code, data, and control flow in a program (Stasko, John., 1998). In the program visualization software the abstractions of the algorithms are graphically represented with the goal of helping users in the process of learning (Byrne, Michael. D., 1999). This technique is developed to help students have a better understanding of the working of distributed algorithms. The program visualization software in this study is designed to help the user to learn two distributed algorithms, termination detection and routing, with the help of visualization. The main goal of this usability study was to evaluate the use of visualization in understanding these distributed algorithms.

This thesis is organized into six chapters. In chapter 2 we provide the background on usability studies, including the goals, stages and methods of conducting these studies. Chapters 3 and 4 describe two usability studies conducted on the web personalization software. Chapter 5 describes the usability study on visualization in learning distributed algorithms. Chapter 6 concludes, and presents findings and a discussion of future work. An appendix is also included, containing the whole package that was given to the subjects, the raw data, and comments collected from users for the studies conducted.

CHAPTER 2

BACKGROUND

Definition: The field of Human Computer Interaction (HCI) is mainly concerned with making systems easy to learn and use. Usability is the key concept in this field. Usability studies are usually performed by the system designers /developers or a usability specialist to determine if the existing system or prototype meets the user's expectations and needs and to inform the continued design and implementation of the system.

The introduction of software into the real world without any usability testing can adversely affect the acceptance of the software by the intended user community and thus affect the success of the project. If software is difficult to use or hard to learn, then the user may give up and stop using it. A thorough usability study on the software before introducing it into the real world can prevent this.

Goals: The most important step in a usability study is to define the goals of the study (Preece 1994). For example: Does the design of the system meet all the needs of the prospective users? Can we make the design better? What are the changes we need to make to our design before we can make it available in the real world?

Establishing these usability goals on a developing system serves two purposes. First, it helps designers to aim for something concrete and to focus on the specific areas that impact the user the most. The results of these studies then drive the user interface design decisions. The second purpose of setting usability goals in advance is to serve as acceptance criteria during usability evaluation, especially toward the end of the design process. Design efforts are iterated with evaluation until the results of the evaluation indicate that the usability goals are finally met. Thus we can say that usability drives both evaluation and the design process, (Rubin 1994).

Types of Usability Study: Two different types of usability goals exist: qualitative and quantitative.

Qualitative usability goals are general, broadly stated goals that, while not easily quantified, may help to guide design. *Quantitative goals* are objective and measurable (Hughes 1999). They may be classified into ease-of-use goals and ease-of-learning goals. *Ease of use* goals focus on the use of the product by users who have had product training and who use it frequently enough to maintain expert performance. *Ease of learning* goals focus on the use of the product by the first time user, the users who are still in the learning process, or infrequent users. Usability testing is most effective when quantitative data is collected, such as the time required to complete a task or a series of tasks (Randall 1998) or the scores for correct answers. In this thesis we performed experiments to collect quantitative as well as qualitative data such as time and correctness of answers as quantitative data, and general comments of users as qualitative data.

According to Lecerof and Paterno (1998) a broader definition of usability is to measure how relevant the system is for the user, the efficiency of the system, the user's attitude towards the system and how he feels about the system. Learnability, safety and reliability of the system are also important considerations.

The goals of the usability study change from system to system and user to user. For example, if a program is used by a child it is very important that it is easy to use. However, for banking systems, efficiency may be the primary consideration.

A number of cofactors should be considered when designing a usability study. These factors include:

- Who are the potential users of the system? Will the tool be used by some specialized users or any user?
- What are the main objectives of this usability study? Are we testing the tool for its ease of use, its complexity to learn, *etc*?
- At what developmental stage is the system? The goal of evaluation performed during the early stage is to predict the usability of the product or to check the design team's understanding of the user's requirement. During a later stage of development the focus is more on identifying the usability problems and improving the user interface. Usability studies are particularly effective in the initial stages of development (*e.g*., requirements gathering, product design), when changing a product's user interface or design is less expensive.
- How much time, effort, and money are we willing to commit at this stage?
- How accurate should the results of the product be?

Techniques: A number of techniques exist for usability evaluations, (Nielsen 1993), (Duams and Redish, 1994). ***Heuristic evaluation*** uses a predefined list of recognized heuristics to identify usability problems. In this method the engineers and/or end-users independently examine the interface and judge its compliance with a predetermined set of usability standards. After independent study a debriefing session is then held with all evaluators, observers and representatives of the design team to brainstorm possible ideas to address the major usability problems, as well as to discuss the positive elements of the interface design. This method is very quick to learn and easy to use, results are available quickly, and it is inexpensive to implement. Since evaluators conduct evaluation independently, the results obtained are unbiased.

The goal of ***pluralistic walkthrough*** is to systematically review the usability of an interface and its flow from a task-based, user-centered perspective while at the same time considering the design constraints. In this method, end users, product developers and human factors engineers evaluate a product from the perspective of the end-user. The evaluators write down sequentially each action they would take when pursuing a designated task. A group discussion then follows, with end-users presenting their information first.

Feature inspection focuses on the feature set of a product. The inspectors are usually given use cases (any common case where they can use different features of the tool) with the end result to be obtained from the use of the product. Each feature is analyzed for its availability, understandability, and other aspects of usability. For example, a common user scenario for the use of a word processor is to produce a letter. The features that would be used include entering text, formatting text, spell-checking, saving the text to a file, and printing the letter.

In a ***formal usability inspection*** usability issues are reviewed within the context of specific user profiles and defined goal-oriented scenarios by applying a task performance model and heuristics. This involves a six-step process, namely Planning, Kick-off meeting (team together for first time), Preparation, Review, Rework, and Follow-up. Every member has his own responsibility. A moderator manages the process. The design owner is responsible for representing and upgrading the product being evaluated. An evaluator finds and reports the usability problems, and another experimenter records all identified problems and decisions. This study provides unbiased findings, lists of usability problems and implemented solutions. Such an evaluation of both cognitive and behavioral tasks, is an efficiently managed process.

Cognitive walkthrough is used to evaluate the ease of learning to use a tool, mainly by exploration. It involves imagining people's thoughts and actions when they use a product interface for the first time (Shneiderman 1998). The method uses an explicitly detailed procedure to simulate a user's problem-solving process at each step, checking to see if the user's goals and memory for actions can be assumed to lead to the next correct action. There are three steps in the procedure (Wharton 1994):

- *Preparatory*: to find out the type of users, tasks and action sequence for each task.
- *Walkthroughs:* can be an individual or group process.
- *Analysis*: where the results are analyzed.

Empirical methods involve data collection through an experimental test and the evaluation of that data to prove or disprove a hypothesis. In this method, a hypothesis is posed depending on a set of objective measures for the evaluation. A plan for how measures are collected is determined and then the subjects for the test are found and the data is collected and analyzed to determine whether the proposed hypothesis is correct or not. This study is effective for establishing cause and effect and for addressing a specific question or problem through focused testing.

The studies mentioned in this thesis also take a similar approach, where users are given specific tasks and the results obtained are used to evaluate the software or the tool. Both the studies first determined the objectives for the evaluation, the data to collect, and the criteria to recruit users. Then, the subjects were recruited, primarily graduates and undergraduates from the Computer Science department at the University of Georgia. Pilot studies were conducted before the actual experiments. Each study went through two runs of experiment.

Multiple methods exist for collecting data. An essential element of these methods is that no feature of important information should be lost and that we can get the best response possible.

Data collection methods: these include the use of instrumented software, usability questionnaires, focus groups, field research, think aloud protocol, and log sheets. Some of these are described below:

a) **Instrumented software:** In this method extra controls are added to the software so that every mouse click and keystroke performed by the user is stored in a binary log file. The time taken to complete each action is also recorded. In this way the analyst can explore the user's interactions in a number of ways such as:

- Types of tasks performed by the user.
- Features frequently or rarely used.
- Order of performing the tasks.
- Time taken by the user to carry out specific tasks.
- Errors made by the users, such as selecting an incorrect menu choice followed by a correct menu choice.
- Frequency of request for online assistance.

b) **Usability questionnaire:** A usability questionnaire is a way of measuring the usability of a system through answers to questions and ratings given by the users. One can collect quantitative as well as qualitative data using this technique (Preece 1994).

This data helps in producing charts and graphs, which provide a visual overview of software usability. In the web personalization study of this thesis the data collected from the questionnaire was used to make some graphs and to calculate data such as the estimated time it will take to develop an application of the same level of complexity as the user developed in the experiment.

The quality of the data collected by the questionnaire depends on how much thought goes into their design and also how sincere the user is in answering the questions.

c) **Focus group:** This is a small group of users who meet to discuss the usability of the system. Usually there is a coordinator who encourages the discussion. The focus group allows users to exchange ideas about the product in a ‘peer to peer’ situation. Results from their discussions are usually presented as a report, which is fed back to the company’s usability group. The informality and group setting of the focus group allow problems and feelings to be aired in a way not possible in other more formal usability test situations. These groups allow direct contact with the users, leading to specific and constructive suggestions.

d) Field research: This is an attempt to put new interfaces in realistic environments for a fixed trial period. This involves going to the client or customer sites to look at the active software and also observe and interview the users. Field research gives a realistic picture of problems and acceptance, as the researchers or the software developers observe the sites after installation of the software.

e) Think aloud protocol: In this method users are required to record their moves either in writing, on audiotape or on video. They are asked to work with the software in a normal way and should feel free to explain out loud what they are doing and if they have any suggestions, thoughts, reactions or problems that they encounter while using the software. This verbal version of the instrumented software has many advantages, for example:

- It lets designers know any wrong move that the user conducts.
- It allows you to see the point where the users hesitate, change their mind or give up, etc.

Usability testing conducted at Lotus is an example of the think aloud protocol, in that users are observed by watching them and hearing their comments while they perform given tasks, Lewis (1982).

f) **Log sheets:** This is another inexpensive and remarkably effective method of collecting data for a usability study. It consists of a sheet of paper with headings, such as Time, Task, Problem and so on. The users record into these sheets whenever they encounter any problems. These sheets are collected daily and are used to provide feedback to the developers on any usability problems.

g) **Usability Laboratories:** These laboratories can vary from a highly equipped resource, full of hardware and software, to a simple workstation in a room set aside for testing. Usability-laboratory staff participate in design reviews, provide information on software tools and help to develop the set of tasks for the usability test. The detailed test plans are developed many days before the usability test, which comprises of the list of tasks, subjective satisfaction and debriefing questions. The number, types, and source of participants are identified and pilot studies are conducted a week ahead of the real testing. During the usability study the participants are told what they will be doing and how long they will be expected to stay.

Participants in this study should always be volunteers and should fill out a consent form before starting the test. During the testing users are invited to think aloud about what they are doing and after some time period they are asked to make general comments or suggestions. Two examples of this type of laboratories are the IBM Usability Center in the UK (Shneiderman 1998) and a usability laboratory at Microsoft, in Redmond, Washington (Reed 1992).

h) **Online or telephone consultants:** Online and phone consultants can provide effective and personal assistance to the users experiencing difficulties with the software. These consultants are an excellent source of information in the problems users have and can provide suggestions for improvement and extension of the system. Example: America Online has live chat rooms for discussion of user problems.

i) **Online suggestion box or trouble reporting:** This type of testing involves electronic mail services to allow users to send messages to the designers or developers. Such online suggestion boxes encourage users to make productive comments, used towards improvement of the system.

There are many more good evaluation techniques, which are not discussed here, such as online bulletin boards or newsgroups, user newsletters and conferences, and competitive usability testing.

CHAPTER 3

EVALUATION OF SOFTWARE USED TO CREATE A PERSONALIZED WEB PAGE

Introduction: Personalization of a web page involves dynamically altering the contents of the web pages according to the preferences of customers or clients, so that each client gets information specific to his needs. Managers of public and private firms use software to personalize web pages for their clients. These personalization technologies can be used to increase the efficiency and effectiveness of a manager's work by enhancing transaction processing and the delivery of services (Wells *et al*., 2000) as well as save clients time by providing only the information that meets their needs (Smyth and Cotter, 2000). Design of the personalization of the web pages may depend upon the status, condition, environment, interests, behavior, and other characteristics of the client (Mobasher *et. al*., 2000).

This study involves the evaluation of software to be used by managers in the government sector to create personalized web pages based on the needs of their clients. Two different types of personalization applications can be developed using this software: ***Occurrence Type*** and ***Priority Type***. In the Occurrence type the users receive messages based on the user's profile (set of characteristics as determined by the way the user has answered a set of questions). The Priority type application provides all users with the same set of messages, but the messages are in different order depending on the user's profile.

There were three main goals in this study:

1. To identify the elements of the software system that need improvement, and to suggest to the creator ways to improve the software.
2. To identify the elements of the training material that need improvement and find ways to improve them.
3. To evaluate the user's ability to create personalized web pages using this software "out of the box". That is, to determine if, and to what extent, users are able to learn the software on their own, without individualized instruction.

Experimental Procedure: Forty students from a Human Computer Interaction (HCI) class offered in the spring semester of 2001 at the University of Georgia in the department of Computer Science participated in the study. Appropriate approval was obtained from the human subjects office at the University of Georgia. The majority of the students were majoring in Computer Science. The study was conducted using PC computers running Netscape Navigator (version 7) or Internet Explorer (version 4 and above).

Before starting the usability study the users were asked to sign a consent form, which contained information about the purpose of the study, the time allotted to perform the tasks and the kind of tasks to be performed. The consent form also stated that the goal of study is to evaluate the usability of the software, rather than to judge the performance of the participants. Finally, the subjects were reminded that their participation is voluntary. The users were randomly assigned to one of two groups, each group containing approximately 20 students. One group evaluated the software for its use in creating the occurrence type of applications and the other group evaluated it for creating the priority type applications. Students in each group were asked to perform two levels of tasks, one simple and the more complex. The groups were further divided into subgroups containing a maximum of four subjects and one supervisor. The students were given 75 minutes to go over the training material (appendix A, page 56), understand the tasks they were to perform, and actually perform the tasks. The training material contained a full description of the experiment, the step-by-step procedure the subjects were to follow to complete the exercises, some examples showing those steps, and the actual exercises to be performed. The tasks were to create two different web applications using "marketing reports" (the exercises describing the high or low risk of a condition a person has depending upon the factors affecting those conditions) developed specifically for this study. One marketing report was for an emergency management advisor web site and the other was for a public health wellness improvement web site (appendix A, page 56). To create the websites the students were asked to develop questions or statements based on the conditions described in those marketing reports and then specify the logic through which to associate messages with each condition. The supervisors were advised to merely distribute the handouts with the directions and the explanation of the task and let the user go through the directions and tasks independently, in order to accurately evaluate the "out of the box" usability of this software.

Users were also provided with a questionnaire, which they were asked to answer after completing the experiment (appendix A, page 55). Additional information about the subjects, such as computer skills, SAT scores, gender, and grade point average was also collected at this time.

The training material provided to the group developing occurrence type applications was different from that provided to the group developing priority type applications. The examples provided and the strategies to build logic on the statement/questions were different for the two applications. Specifics of each group's procedure are described below:

Occurrence type group: In this group, the user's first task was to figure out the questions that needed to be asked to develop the application. The next step was to attach messages to the questions and set up the logic for which text message would be returned in response to each question. The most important thing for the user was to make sure that all the questions were logically associated with text so that no matter how a web client's profile was configured, the client would receive a single message for each item. For example, the users were asked to set some type of logic with all the messages so that, depending on the client's answers, they will receive a message showing either high or low risk of a particular outcome (e.g., sudden illness, disease, a dangerous situation, taxation etc.). For example:

(1) *Simple text-logic relationship:*

Marketing Report: People who smoke are at higher risk of having cancer.

Condition Statement: You smoke very frequently.

Scale:Very False 1 2 3 4 5 Very True

Example of setting logical text with the condition:

Logic	*Text Message*
Smoking condition is equal to or greater than (=GT) 4.	You are very likely to have cancer
Smoking condition is equal or less than (=LT) 3.	You are not likely to have cancer.

(2) *Complex two factor relationship:*

Marketing Report:

People who smoke are at higher risk of having cancer.

The older a person is the more likely he is to have cancer.

Condition Statements:

You smoke frequently.

Scale:Very False 1 2 3 4 5 Very True

You are old.

Scale:Very False 1 2 3 4 5 Very True

What message is returned (high or low risk of cancer) is determined by set of logic statements shown in the following table:

Text Message	***Logic Settings***
1. HIGH RISK: You are very likely to have cancer.	1. Smoking condition is equal to or greater than (=GT) 4. Old condition equal to or greater than (=GT) 3. 2. Smoking condition is equal to or greater than (=GT) 3. Old condition is equal to or greater than (=GT) 4.

2. LOW RISK: You are not likely to have cancer.	3. Smoking condition is equal to or greater than (=GT) 1. Age condition is equal to or less than (=LT) 2. 4. Smoking condition is equal to or less than (=LT) 2. Age condition is equal to or greater than (=GT) 1. 5. Smoking condition is equal to (=) 3. Age condition is equal to (=) 3.

No matter how the user answers these questions he should receive exactly one message (high risk or low risk) per condition, the message that best suits his or her profile. In the occurrence type of application, as the number of testing factors increases, the complexity of the logic pattern to determine the end result also increases. The developers of occurrence type applications must be very careful when setting the logic, to avoid duplicate responses. Many more complex text logic patterns are described in the training material given to the users, and can be seen in appendix A, page 55. The last step in the development of the application was to test the application for its correctness. The users had to answer a set of questions they developed and check if they received the right response.

After the students completed the experiment, they were asked to fill out a questionnaire consisting of 12 questions. Some of the questions were to rate their agreement with a statement on a scale of 1 to 5 and some questions solicited general comments about problems, usefulness and ways to improve the application. Users were asked to fill in the start and stop time for both the tasks and to indicate the type of the application they built. Subjects were also asked to rate their experience with learning new software, their perception of their ability to teach others to use the application, their difficulty in analyzing the exercise problem to create the logic, their difficulty in using the software to implement the logic, and the clarity of the directions. Information about problems experienced while developing the application, opinions on useful design features of the software and suggestions to improve the usability of the application were also collected. The detailed questionnaire is included in the appendix A, page 54.

Priority type group: The experimental procedure for the priority type group was similar to that of the occurrence type group. The primary difference was that the training material described the strategies to perform priority type applications. The resulting web page created by using this application shows all the messages associated with the text but what matters in this application is the order in which they appear. That is, the subjects developing the priority type applications were involved in determining in what sequence or order a series of messages would appear, and this order was guided by the information in the marketing reports. The sequence in which the messages appear can be affected by a number of contributing factors. For some issues in the report, there will be only a single factor that is reported as affecting a person's risk or interest and hence only one priority value. For other issues, multiple factors will contribute to risk or interest. In the case where multiple factors make a contribution, the contribution to risk/interest is apportioned equally to each factor. The developer of the application has to concentrate on the relationship between a message and the priority that the message should have in relation to the other messages in the set of the messages to be displayed. Unlike the occurrence type, the users do not have to worry about eliminating duplicate messages. This can be explained in more detail using the following example: For job success the contributing factor is school grades, and for success as a wrestler there are two contributing factors, age and body weight. A person would be of equal risk of job success and wrestling success only when all three factors (school grades, age and body weight) are of same value on a 5-point value scale. If the school grade factor and age factor are rated 3, but body weight is rated at 4 on the scale, then the wrestling success message will appear before the job success as the total value for the wrestling factor is 3.5 (3 (for age) + 4 (for body weight) /2) which is higher than the value of job success (3). Detailed examples on priority type applications are described in the appendix. Again, the subject was expected to test the correctness of the result, and complete the questionnaire.

Results: Data collected includes the time to complete the task and the artifact created by the subject. The experimenter then evaluated the quality of the artifact (website) produced. Users of each group were provided with a grade ranging from 0 to 10 for both exercises based on the correctness of the tasks. Results are summarized in table 1(raw data included in appendix B, pages 67 and 68). The correctness was measured based on the type of questions asked, type of messages associated with the questions, correctness of message-logic pairs, and completeness of the task.

According to table 1 there was no statistically significant difference between the mean scores for exercise 1 and exercise 2 for both the application types occurrence and priority. Table 1 suggests that the subjects who worked with the priority type applications scored somewhat higher on both the exercises than those who worked on the occurrence type, but the difference was not statistically significant.

Table 1: Average scores obtained by the user for exercise 1 and 2 for Occurrence and Priority type applications.

	Occurrence		*Priority*	
	Exercise 1	Exercise 2	Exercise 1	Exercise 2
Mean	**3.4**	**3.1**	**4.2**	**3.2**
Std	**3.0**	**3.2**	**3.6**	**3.7**
Variance	**9.2**	**10.1**	**13.1**	**13.7**

The time required to perform exercise 1 and exercise 2 for both groups was calculated and analyzed. Figure 1 shows the time each user took to perform task 1 (solid line) and task 2 (scattered line) for occurrence type applications and Figure 2 shows the same for priority type applications. Almost all the users for both types of applications took more time to complete exercise 1 than to complete exercise 2.

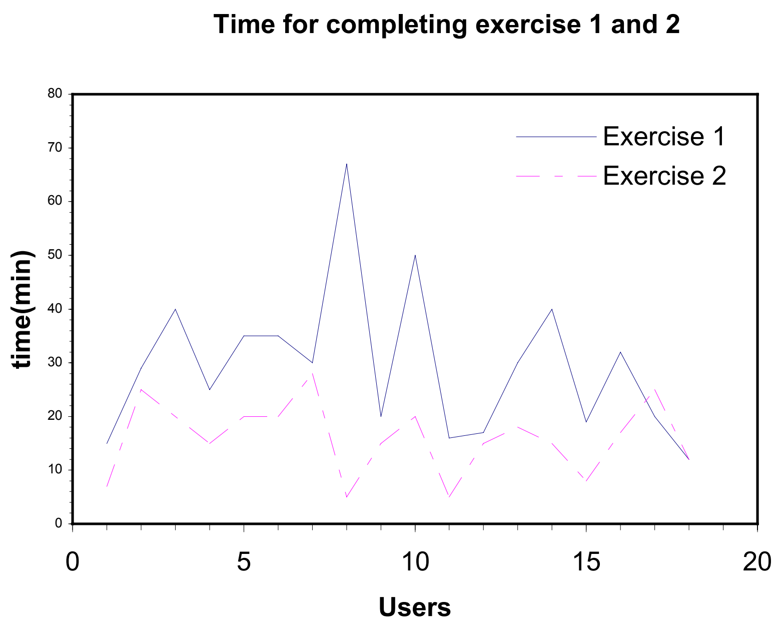


Figure 1: Graph showing the time required to complete task 1 and task 2 for the users of occurrence type of applications.

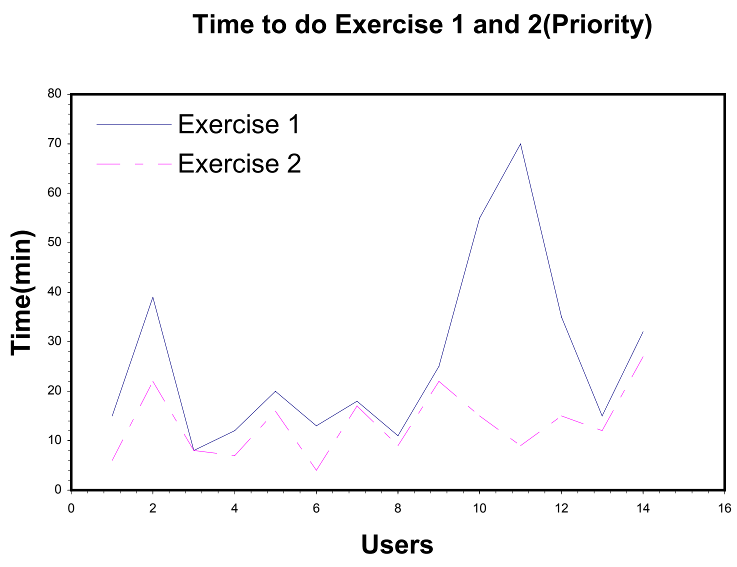


Figure 2: Graph showing the time required to complete task 1 and task 2 for the users of priority type of applications.

The average time, standard deviation and the variance for both exercises for each group are presented in table 2. Mean values for exercise 1 for both applications are greater than exercise 2, though they were not statistically different. The times required to complete exercise 1 and exercise 2 for occurrence type applications were also compared with those of priority type applications, as presented in table 2 and graphed in figure 3. The average time required to perform both the exercises for the occurrence type application was greater than the average time to complete the exercises for the priority type, but again the difference was not statistically significant.

Table 2: Time required to perform exercise 1 and exercise 2 for both Occurrence and Priority type of applications.

	Occurrence		*Priority*	
	Exercise 1	Exercise 2	Exercise 1	Exercise 2
Mean	**29**	**15.7**	**27.2**	**14.1**
Std	**14.7**	**6.9**	**18.7**	**6.8**
Variance	**216**	**46.9**	**348.5**	**45.9**

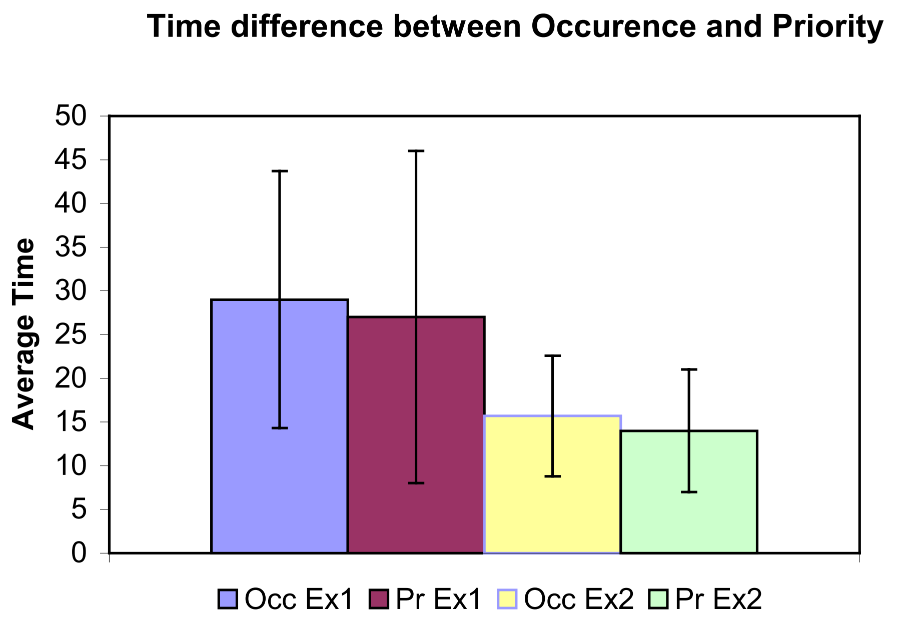


Figure 3: Comparison of average time required to perform both tasks using occurrence type applications with those of priority type applications.

Users were also asked to estimate the time it would take them to perform another task of same difficulty after this experiment. The actual fraction of time required to perform exercise 2 after completing exercise 1 was calculated and compared to the estimated time. The correlation between the two for the occurrence type of study was 0.51 and that for the priority type was 0.133, as seen in figures 4 and 5 (the correlation was calculated using EXCEL). These numbers determine that the times (the time required to complete exercise 2 and the time estimated to complete another exercise) for both the applications were not highly correlated. Subjects estimated that it would take them an average of 50% (occurrence type) and 42% (priority type) of the time originally required to complete an exercise of similar difficulty.

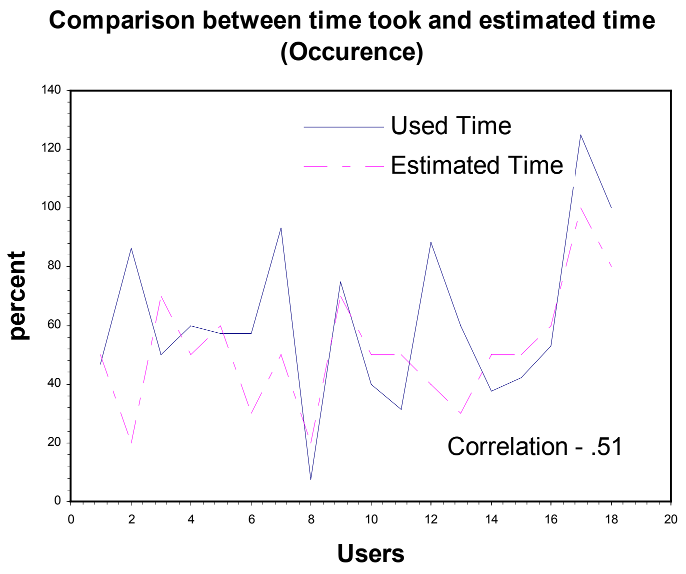


Figure 4: Graph showing the correlation between the actual percent of time taken by the user to perform exercise 2 compared to exercise 1 and estimated time it will take the user to perform another exercise of same difficulty for occurrence type applications.

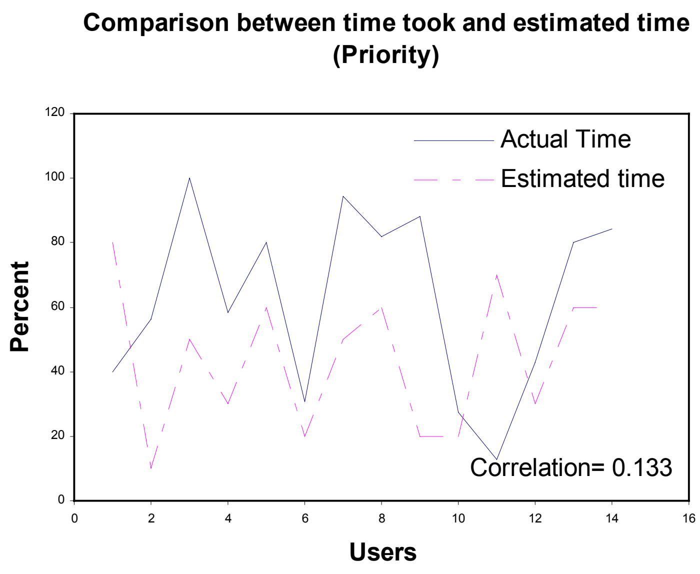


Figure 5: Graph showing the correlation between the actual percent of time taken by the user to perform exercise 2 compared to exercise 1 and estimated time it will take the user to perform another exercise of same difficulty for priority type applications.

Users remarks were also collected and are summarized as follows:

Comments about problems experienced: According to some people the instructions were unclear, hard, confusing and tedious to read and understand in the given amount of time. They had problems understanding the tasks, deciding where to start and the kinds of questions to ask. They also had trouble understanding the results and goal of the application. According to them the buttons they were asked to press in the instructions were hard to locate due to the structuring of the text blocks page.

Comments about useful or helpful design features: Adding messages and attaching the logic with the messages was easy, intuitive and well designed. Buttons, text fields and revision functions were helpful and well placed. All the components were highly visible and help messages were useful. The users liked the testing feature in the software, where they could determine the correctness of the logic and could go back and correct it if the logic was wrong. Pop-up windows to confirm the logic decisions were also helpful. With smaller tasks or simpler problems the logic was easy to use, but with larger problems users felt it would be very difficult. After finishing the first exercise the second exercise seemed much simpler. This suggests that once you are familiar with the software, it is much easier to use. The main control panel and the feedback on logic is good but could be improved, and the overall graphical user-interface was easy to use.

Comments about improving the usability of the application: The main suggestion was to provide simple, clear, concise and step-by-step instructions. In the current software when they create or change a text block, they have to submit each text block separately, so the user should be able to add all the text blocks at once and in the end click the submit button for all the changes made. The use of multiple sections and colors on one control page made the screen seem busy. One way to simplify the screen would be to divide the control across multiple pages; this will help improve clarity and reduce the numbers of fields the user can select. For a non-technical field graduate, logic examples and walkthrough applications would help. On every page there should be links to the control panel and to test and review the logic. The layout of the screen, especially the text logic screens, were hard to pick up at first glance. According to some, the biggest problem was that the software lacks a clear and brief instruction on the interface. Bulleted steps written on the interface and a short tutorial online help are extremely helpful. The instructions need to be rewritten, perhaps with a more complete initial overview and more explanation of what the different fields mean/do for setting the logic.

Discussion: The main goal of this study was to evaluate the ease-of-use and learnability of the web personalization software and also to get some feedback about the elements of the software. In addition to these goals, the effect of directions, tasks and time on the study itself were evaluated. This section discusses the results obtained by analyzing the data collected during the study and various problems identified by the subjects in the software and the study. These results and suggestions will be used to refine the features provided by the tool and make them better in terms of usability. They will also be used to evaluate and improve the tasks, directions and the overall study, which will work toward making the software more usable for the intended users. The average scores and time for both the occurrence and priority type of applications suggest that subjects had more difficulty with occurrence type applications than the priority type. This might be because the developer of occurrence type applications needed to apply a complex logic pattern that determines which user will receive which message, and also to prevent duplicate responses. The priority type applications were relatively easy, and the developers did not have to deal with eliminating duplicate messages. They only had to deal with the priority that the messages should have in relationship to the other messages in the set of messages to be returned. Almost all the students spent more time to complete exercise 1 than to complete exercise 2 for both types of applications as shown in figures 1 and 2. The average time to complete exercise 1 for the two applications was also more than the average time to complete exercise 2 (see table 2). This may be because of the reason that they got familiar with the software and the study by the time they finished exercise 1 and thus could finish exercise 2 faster.

By looking at the average scores obtained by the students shown in Table 1 of the results section we can see that the subjects did not perform well. Several factors may account for this. For example, the directions to perform the exercises were not clear, the software itself was not that user friendly, and bugs existed in the software. We discuss each of these problems in greater detail in this section.

Problems with the software: According to the users there were many different problems with the software. The buttons they were asked to press in the instructions were hard to locate on the web page, because the correlation between the steps on the web page and the steps mentioned in the directions is poor. The current tool has many items clustered on the same page. For example, all the steps from first to last are on the same page. These steps can be broken down so that they come one after another. This will make things easier and the user will not have to find out what step they need to perform next, rather the application would take them through those steps automatically. Thus, the overall structure of the application forms and text block pages need reorganization. Something could be done regarding the position of the buttons, such as having the submit button on the bottom of the application form or on both bottom and top instead of only the top. There were also some parsing errors found during the experiment. For example, the users could not submit questions or messages containing semicolons, commas or quotes. This problem had the potential to distract the user and then cause them to take more time to complete the tasks than they should normally take. All these things can cause the user to lose interest and finally give up and stop using the software. The directions consist of some examples explaining how to set the logic-text blocks but the students do not have that much patience to first go through the examples and then do the exercises. Most of them wanted to get done with it as soon as they could.

Problems with the study: As discussed above, results from table 1 suggest that the users had more problems with occurrence type applications than with priority type applications. Subjects performing tasks to develop occurrence type applications also took more time than those who developed priority type applications. This may be due to the level of the tasks/exercises they had to perform, which became difficult to understand and complex to implement as the number of contributing factors increased. According to the remarks of the students, the directions were not clear and were hard to understand; they were provided with an extensive set of instructions that were confusing. Some subjects lost the overall picture of what the software was designed to do.

Time was also a big issue, as they had to go through the instructions as well as perform two tasks in 75 minutes. The students were also asked to specify the start and stop time for both the exercises but during the evaluation of the results it was found that in some cases students put the start and stop time but did not develop an application. Again, problems with the directions may be to blame. The other reason that users lost their applications may be due to the crash of the system during development or by accidentally clicking the *delete application* button.

After evaluating the results and identifying the problems in both the study and the software we decided to conduct the study again, and the changes described below were made:

*Changes to the study***:** Due to the problems students faced we decided to provide handouts with bulleted step-by-step instructions to develop applications. Extensive instructions were also provided on the site where subjects developed the application. As soon as the user logs in, a simple training task with the step-by-step procedure to create the web page is automatically given. This way the user will know in a short time what they have to do and what they should produce as a result. As the second study focuses more on the time it takes the users to complete the exercises correctly, the supervisor checks the time the subject starts the exercise and enters the stop time only when the exercise is completed correctly. The tasks should also be modified, as according to some users the tasks were unclear and hard to understand.

Suggested changes to the software: In the previous software all the steps involved in developing an application are clustered on the front page. To simplify this, instead of having all the steps on one page, we decided to organize the page so that the first time users see only the minimal number of screens and controls needed to complete a basic application. Optional parameters were either eliminated or put into a place where they are not visible unless the user decides to request them. The whole procedure was divided mainly into 3 step-by-step procedures in a lock-step fashion, first profile questions/conditions, then messages and then adding logic. The hope is that this change will make the revision process easier compared to the previous version of the software. Previously, to revise a text logic block the user had to either know the number of the block he wanted to change or had to choose from a form containing all the test logic blocks the ones he wanted to change. In the new version of the software, users click on a specific message and then get access to the logic patterns that are associated with that message.

To make the software easy to use and understand, the color scheme used on the web page can be changed, so that the buttons are more visible and obvious. We can also work on making legends and labels on the page more explanatory so that they are easy to understand.

Conclusion: The personalization software used in this study was evaluated for its usability and learnability to develop applications to create personalized web sites. This software is developed to be used by managers in public sector, to develop applications to personalize web sites for their clients. The results and the user feedback obtained from this study suggest that the software and the study require substantial changes to make the software easy to use and learn. The main drawback of this study was that the subjects had to perform several tasks in a limited amount of time (around 75 minutes). They had to read through an extensive amount of material, including directions to perform applications, examples, and strategies for doing the applications, and simultaneously perform the exercises to develop two personalization applications. Likely, this was too much work for the given amount of time. The bugs and problems in the software, as mentioned in the results, were also a big distraction for the subjects performing the task. The results and the feedback received from this study will help us in improving the software as well as the directions, so that we can reach our main goal, making the software easy to use and learn. After making all the changes to the study and the software, another usability study was conducted on the software. This second study was more controlled and focused more on the correctness of the task rather than time required to complete the task. The second study is described in chapter 4.

CHAPTER 4

SECOND USABILITY STUDY OF WEB PERSONALIZATION

Introduction: An evaluation of the results of the initial user study provided suggested changes and other feedback from the subjects. The software was revised in accordance with these results. However, it is important to repeat the usability study to assess the improvement in the usability of the software. A repetitive, cyclical design process lends itself easily to user-centered design; it is useful for designers and developers to revise the design iteratively through rounds of testing. The main aim of the repeated usability study should be to compare the first and second study to see whether the new version is more usable than the old version. The number of times a usability study should be repeated depends on various factors, including planned time-to-market of the software, cost considerations, and user's expectations of the software. For software to survive in the world of users it is crucial that it fulfill the expectations of its users. The usability study helps us to do that. Repeating the usability study also helps in improving the usability study itself, for example by changing elements such as directions, questions asked, and specifics of the tasks.

The second usability study was divided into four parts. The first step involved working with the developer of the software, John O' Looney, to modify the software and make almost all the changes recommended by the subjects in the previous study. The second step was to modify the usability study plan based on the recommended changes, modify the directions, create a practice test and modify the exercises. Third was to execute the usability study, and the last step was to analyze the collected data. The main goal of this usability study was to make recommended changes to the software and to improve the software in order to make it as usable as possible. The usability of this software was again evaluated for its usability and learnability after all the changes were made to the software, directions and the study.

Experimental Procedure: The comments and the results from last experiment were used to make changes to the study and the software, as follows:

- The main page of the software was divided into many pages, unlike the previous version of the software, in which all the steps were on the same page.
- The directions were made simple and small. They contained the step-by-step procedure for both a sample exercise and the real exercise (appendix A page55).
- Before starting the experiment, the software and the directions were thoroughly tested for bugs and to confirm the flow of the steps for creating the personalization application.

A total of 20 students were recruited from the department of computer science, 15 graduate students (13 males and 2 females) and 5 (4 males and 1 female) undergraduates**.** The subjects used in this experiment were different from the subjects used in the previous experiment to eliminate biased results. The experiment was conducted using Internet Explorer on PC's. Evaluation of this experiment was based on the time it takes a person to learn the web personalization software. For the results to be included in the study the subject was required to stay until he or she finished the task correctly, but no longer than 75 minutes. Before they started the experiment, subjects were asked to sign a consent form specifying that this experiment was to evaluate the personalization software and not their ability, that they did not have any mental or physical risk while performing this experiment, and reminding them that they were free to stop at any time without completing exercise (No one left early) (see form page 53, appendix A). The subjects were asked to perform a practice task before doing the actual task. A list of bulleted steps to perform the practice task as well as actual task was provided with the exercises. For reference, the extensive directions were included on the web page where the software was located. The directions were also accompanied with a questionnaire, which subjects were asked to fill out when they completed the study. They were also asked to record the stop and start time of the practice exercise and the real task on the questionnaire. Unlike the previous study, in this experiment the subjects were asked to build only the *occurrence* type of application, due to less availability of subjects and tasks. The subjects were asked to create questions/statements depending on the issues mentioned in the task (included in the appendix A, page 61), create appropriate messages accordingly and then build the logic to attach the messages to the questions/statements (as explained in detail in the previous chapter). These steps are the main steps to be performed by the target users while developing the software. At the end, subjects were also asked to test their application. The logic of the task for each subject was tested against the correct logic for the task and if the subject's logic was not totally correct, the subjects were asked to go back and correct their logic. This process was repeated until their logic was completely correct or the total time they required was over 75 minutes. One student was picked randomly from among the list of students who participated and was provided with a prize of $20.

Results: This experiment was conducted to evaluate the learnability of the software and thus the time required by the students to complete the exercise correctly was recorded. As time was not a limitation in this experiment and the results of the subjects were checked as they finished and subjects were asked to correct their work if something was incorrect, 70 percent of the subjects scored 10 on a scale of 0 to 10. The maximum and minimum time required to complete exercise 1 correctly was 46 and 20 minutes respectively with an average of 32 minutes (Table 3).

The average score obtained by the subjects in this usability study was 8.5 with a standard deviation of 2.5 (Table 3), raw data included in the appendix (B, page 69).

Table 3: Average score obtained and average time required to complete the exercise by all the students as well as those who received a perfect score.

	Scores	Average time to complete exercise correctly	Time to do real exercise
Average	8.5	32	32
Standard deviation	2.5	7.6	10.4

The students were asked to perform a practice run before they start the real exercise to permit them to have a good understanding of the software. The students were taken through all the steps to complete the sample exercise using a set of directions. Then they were asked to do the same for the real exercise. Table 3 shows the average time required by the subjects to do the practice and the real exercise.

The subjects were asked to comment about the problems with the software and the ways to improve it. These comments are described below:

Problems with the software: Students commented that because the directions were provided both on screen and on paper, they were not sure which to follow. Some students mentioned that the step-by-step directions were very detailed and a little confusing; they were not always sure where to go. Getting the logic pattern right was a challenge for some subjects. One student found that the feature permitting the addition of logic from the review page was not working and he wasted some time trying to make it work.

Helpful design features: Students really liked the feature that they can revise as many times as they want and test their logic. They could also delete and modify logical links. The whole idea of creating logic and assigning values was very interesting to them. According to some subjects the logic sets were neatly broken down and were very objective. The overall layout, the color combination and graphical user interface was good and the handout was also very clear.

Suggestions to improve usability of the application: Some subjects felt the need of some introduction about what this software is about. With the directions provided it was hard to judge if the application was user friendly or not, if they did it right, because they followed the directions on the handout not the directions by the application. Some subjects suggested giving the user a matrix or some kind of a flowchart with all possible values of responses so that user does not have to worry whether all conditions are taken care of or not. They also recommended giving some more complex examples for building the logic. There were some suggestions about allowing the user to return to the review logic page and the test page from all the pages and having more detailed directions about the use of the different button selections. They thought that the pages were a bit crowded with information and noted that subjects did not spend time to read that information.

Discussion: As mentioned in the results, around 70 percent of the students received a perfect score of 10. This may be due to many reasons. A primary reason is that the results of each student were tested when they were done and subjects were asked to correct their work if their logic was incorrect. They were allowed to take as much time as they wanted and they were not allowed to leave before they either got it right or 75 minutes had elapsed. Another reason for the improvement in the scores was due to the practice exercise. This exercise was designed to take students through all the steps for creating an application and then testing the logic. This exercise included the exact questions/statements and messages the subjects had to fill out in order to develop that particular application and gave them an idea about the type of questions and messages they had to develop for the real exercise. It made them think logically and find the ways to make their logic correct. Another reason for their completing the exercise correctly could be that they were told that their answers will be checked after they are done and that made them do the exercise more sincerely. The quantitative results (scores and time) from this study could not be compared with the results of the previous study because no subject could complete the tasks correctly in the previous study. Comments collected in this experiment suggest that the software this time was much easier to use, but we are not sure if this is because of difference in the comments or due to difference in the users.
In the older application all the steps to develop an application were included on one control page, shown in Figure 6. In the new version of the software, the steps in creating the application were divided into different pages where one step led to another as shown

in Figure 7, 8 and 9. This feature helped students to be on the right track and go with the flow of steps; they did not have to think what they had to do next. The used directions in this experiment were also short and bulleted.

Directions in bullets were also provided on each page containing a specific step to perform the application, so if the students did not want to look at the directions on the paper they could perform that step by reading the direction on the web page itself. As mentioned above, the applications were tested for correctness with the subjects present and subjects were asked to redo their logic if it was wrong. Most of the subjects did the exercise correct in one run and only a few of them took two to three runs to complete it correctly. In spite of all these improvements, several subjects still could not complete the test correctly: it took them a great amount of time to understand and do the sample exercise and by the time they started the real exercise it was past one hour and they were not forced to stay after that.

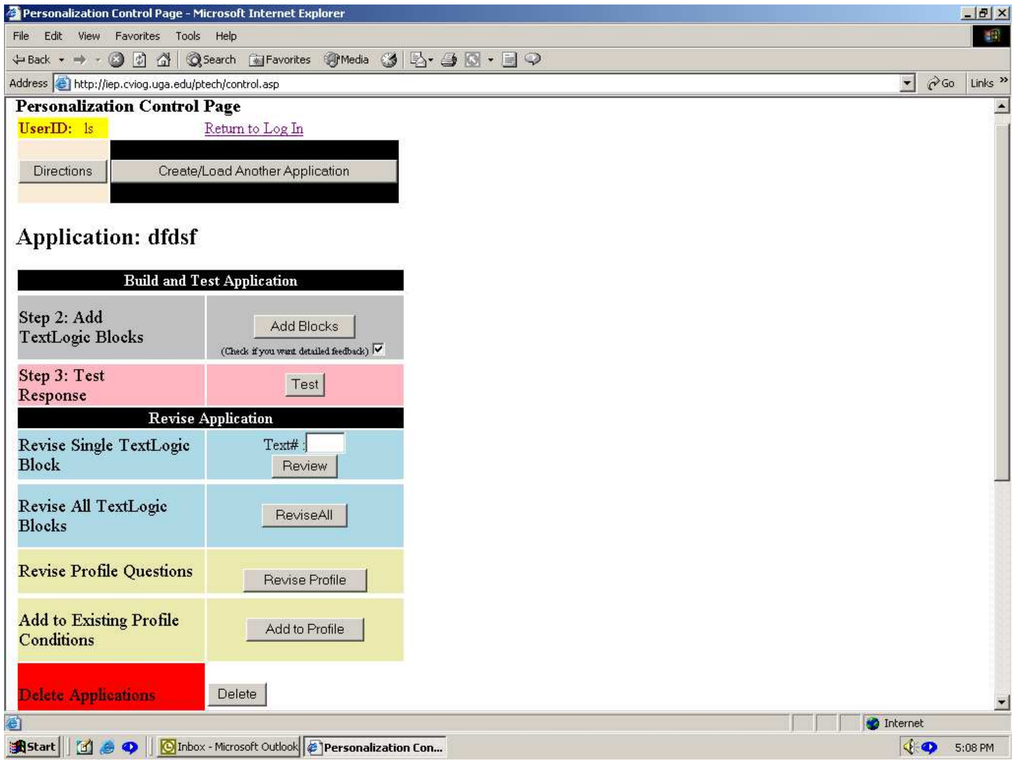
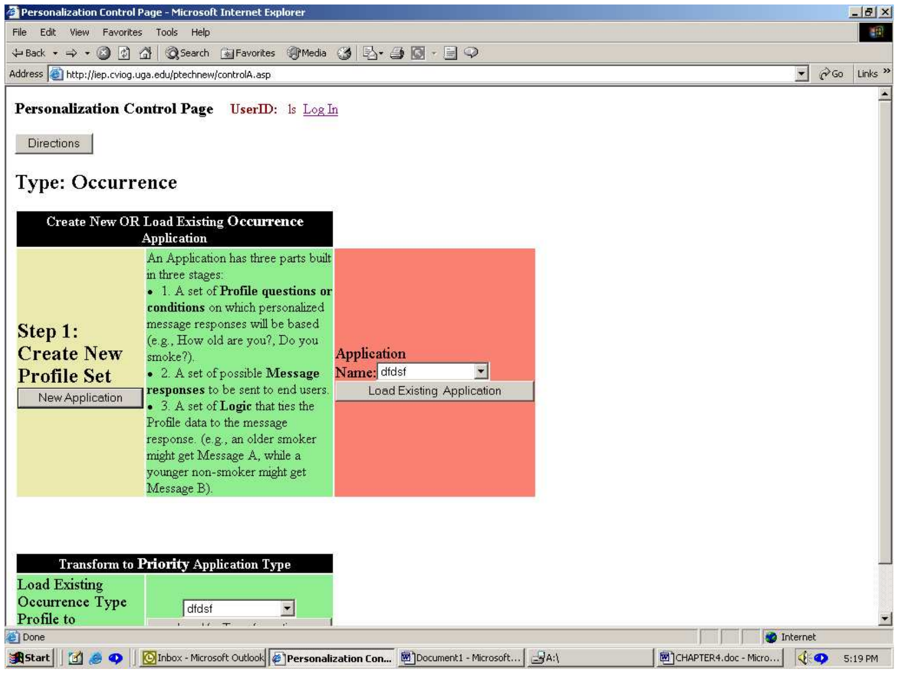

Personalization Control Page - Microsoft Internet Explorer
http://iep.cviog.uga.edu/ptech/control.asp
Personalization Control Page
UserID: ls
Return to Log In
Directions
Create/Load Another Application
Application: dfdsf
Build and Test Application
Step 2: Add TextLogic Blocks
Add Blocks
(Check if you want detailed feedback)
Step 3: Test Response
Test
Revise Application
Revise Single TextLogic Block
Text#
Review
Revise All TextLogic Blocks
ReviseAll
Revise Profile Questions
Revise Profile
Add to Existing Profile Conditions
Add to Profile
Delete Applications
Delete

Figure 6: Snap shot of older version of the web personalization software.



Personalization Control Page - Microsoft Internet Explorer

Address http://iep.cviog.uga.edu/ptechnew/controlA.asp

Personalization Control Page UserID: ls Log In

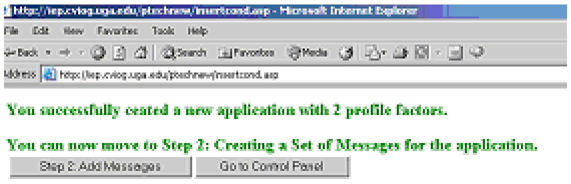
Directions

Type: Occurrence

Create New OR Load Existing **Occurrence** Application		
Step 1: Create New Profile Set New Application	An Application has three parts built in three stages: • 1. A set of **Profile questions or conditions** on which personalized message responses will be based (e.g., How old are you?, Do you smoke?). • 2. A set of possible **Message responses** to be sent to end users. • 3. A set of **Logic** that ties the Profile data to the message response. (e.g., an older smoker might get Message A, while a younger non-smoker might get Message B).	Application Name: dfdsf Load Existing Application

Transform to **Priority** Application Type	
Load Existing Occurrence Type Profile to	dfdsf

Figure 7: Snapshot of step 1 (create new application), in the new version of web personalization software.



http://iep.cviog.uga.edu/ptechnew/insertcond.asp - Microsoft Internet Explorer

Address http://iep.cviog.uga.edu/ptechnew/insertcond.asp

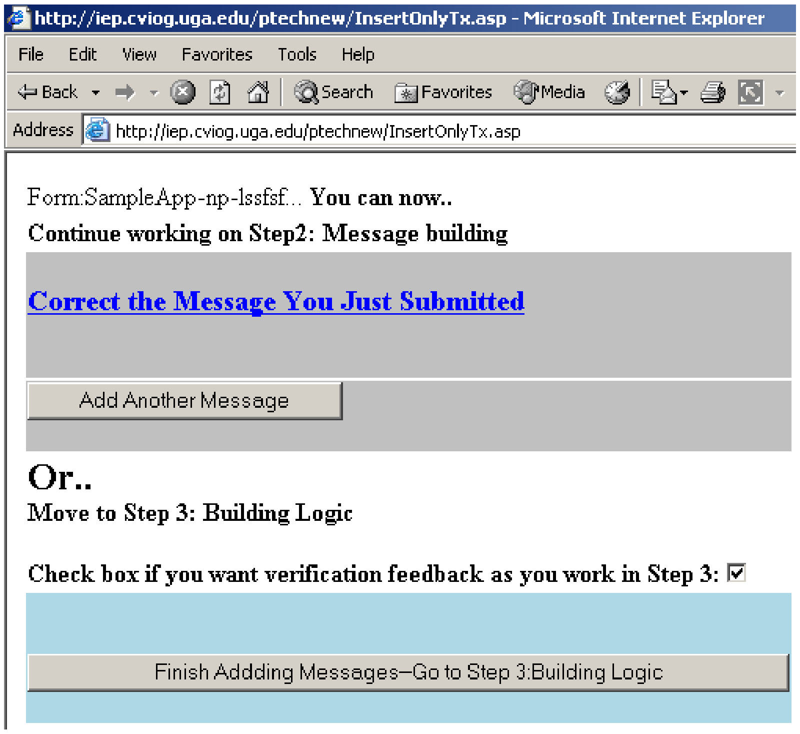
You successfully created a new application with 2 profile factors.

You can now move to Step 2: Creating a Set of Messages for the application.

Step 2: Add Messages | Go to Control Panel

Figure 8: Snapshot of step 2 (Add message), in the new version of web personalization software, which appears after the user is done with step1.

According to the comments made, this software still requires revisions before being ready for use “out of the box”. Based on the maximum time required (46 minutes) to do the exercise correctly, it can be suggested that an exercise of this level is doable and learnable within an hour, whereas it is hard to do two exercises within the same amount of time.


http://iep.cviog.uga.edu/ptechnew/InsertOnlyTx.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://iep.cviog.uga.edu/ptechnew/InsertOnlyTx.asp
Add Another Message
Finish Addding Messages--Go to Step 3:Building Logic

Form:SampleApp-np-lssfsf... **You can now..**
Continue working on Step2: Message building

Correct the Message You Just Submitted

Or..
Move to Step 3: Building Logic

Check box if you want verification feedback as you work in Step 3: ☑

Figure 9: Snapshot of page, which leads to step 3 (building logic) of building an application.

Although we can easily create simple applications where only one factor or two factor logic is involved, it is very hard to create applications where many conditions (more than three of four) and questions/statements are involved. The software should have some kind of capability where it can build the logic by itself or at least assist in thinking logically in some way.

Conclusion: The results presented in this chapter shows that this software is much easier to use and learn compared to its previous version because many users were able to complete the same task correctly in the second study. Based on the average scores and the comments we can say that changes made to the software and the study helped the subjects to perform well. The main recommendations to the developer would be to have more clear directions on the pages of the software, as presented in the handout, and the use of a graphical representation that covers all the cases required to build the logic for personalization. This research shows that for this software to be successfully used “out of the box” by its target user group, the developers must implement some demo exercises that use different levels of logic, so that the user has the opportunity to become familiar with all the steps involved in creating the personalized application, as well as be able to refresh his logical thinking. The software still needs to go through several stages before it could be ready to be used commercially, since the main goal of developing this software is to be able to use it “out of the box”, which means that no training should be involved prior to using it.

CHAPTER 5

CAN VISUALIZATION HELP STUDENTS TO LEARN DISTRIBUTED ALGORITHMS?

Introduction: Increases in technology have provided new developments in the area of graphics and visualization. Such graphical representations of data have the potential to aid in the cognitive process. In the field of visualization, researchers and developers have created many types of visualizations or visual depictions of information (Card, Mackinlay & Shneiderman, 1999). Program visualization is a technique designed to help programmers understand the state and behavior of a program. Roman and Cox (1992) in their paper defined visualization as a mapping from programs to graphical representations.

Program visualization tools provide a continuous graphical representation of the state and the progress of the program as it executes. Tuning and debugging are typical uses of program visualization. Various studies have discussed the importance of visualization in understanding a program, such as the use of graphical visualization for better performance evaluation and understanding of execution of parallel discrete-simulation systems (PDES) (Carothers *et al*, 1999). Topol *et al.*, (1998) discussed the use of two phase visualization (PvaniM) to provide insight into the behavior, efficiency and operation of distributed and parallel programs in a network computing environment. Stasko and Kraemer, 1993, discussed the importance of and developed a methodology to create application-specific visualizations of parallel programs.

Although many researchers believe that visualization is helpful in understanding distributed algorithms, the results obtained from user studies are inconclusive. Most of the studies were based on general observations and interviews done on the students as mentioned in the study performed by Storey *et al*. (1997) to compare three visualization systems for browsing program code and exploring software structure. In some cases the tool used itself hindered the uses progress. Mulholland, (1998) studied the effectiveness of four prolog program visualization systems. The study resulted in mixed up and inconclusive results due to the misunderstanding of the traces made by the tracers. Evaluation of algorithm animations used for learning was done by Byrne et al, 1999. They suggested that one way animations may aid in the learning of procedural knowledge is by encouraging learners to predict the algorithm’s behavior, but no significant results were found. Kehoe et al., 2001 focused on understanding the use of animation and other learning materials by learners to understand a new algorithm. This study was not very controlled and thus not many definite results were found. Zernik *et al*. (1992), used visualization tools in the understanding execution of sophisticated concurrent programs. Brown and Sedgewick (1985), present different techniques for animation for algorithm execution that allow students to explore the algorithm’s behavior by changing data and parameter values and observing the resultant behavior.

A study was conducted by Rong Wu, Mihail Tudoreanu and Ritu Dhawan under the supervision of Dr. Eileen Kraemer. The main aim of this study was to see if visualization can help in understanding and learning of distributed algorithms. A study was conducted to determine the effect of visualization on understanding and learning two distributed algorithms: The Dijkstra-Scholten Distributed Diffusing Termination Detection and

Distance Vector (Bellman-Ford) Routing Algorithm. Two experiments were conducted to evaluate the usefulness of visualization on the user's ability to learn about the above-mentioned algorithms. The second experiment was a rerun of the first experiment after modifying the code, visualization and the user interface, based on the user's feedback and the results obtained from the first study. The visualization, pseudocode, output and the questions were all embedded in the user interface. This chapter mainly focuses on my contribution to this study. A brief summary of the experiment, results and the conclusions of this study are also presented.

Elements: There were three main elements in the design of the study:

- The choice of algorithms
- The design of the visualization
- The creation of questions for each algorithm

My Contributions:

- Implementation of the algorithms:

The algorithms were selected based on the fact that they were interesting and have appropriate complexity for the participants. Both the algorithms were implemented in Java and the distributed systems were simulated using threads. These algorithms are briefly described below:

Dijkstra-Scholten Distributed Diffusing Termination Detection (Dijkstra and Scholten(1980):

This algorithm is designed to allow a node to determine if a task that is computed by multiple nodes has finished or not. The problem is based on how the initiator of the project detects the project's completion. The initiator starts sending the messages to other nodes and then either these nodes acknowledge back or send messages to more nodes. The termination is detected when all the nodes send acknowledgment back to their parents and the initiator (root), which mean all the nodes have finished the processing.

Distance Vector Routing Algorithm:

The second algorithm dealt with the updating of network routing tables. Each node was involved in keeping a table containing information about shortest distances to other node to which it is directly or indirectly linked. The pseudocode and detailed description for both the algorithms is attached in the appendix (appendix C, page 70).

- Participated in the discussion and meetings to develop the study.
- Implemented the demo visualization program (about finding the greatest value in an unordered array) used in the second. This was embedded in the same testing environment as the one used for the formal study.
- Provided the output and the pseudocode of the algorithms to be incorporated in the real study environment of the formal. This was used to make students familiar with the study environment.
- Helped in developing the tasks used in both study.

Summary of procedure and results: Two studies were conducted on graduate and undergraduate students in the department of computer science. The students were asked to answer some questions about the algorithms mentioned above either using visualization or using the code and the output. The first study used both the algorithms mentioned above. The second study used only the Dijkstra Scholten algorithm due to a smaller number of subjects and the limited time available.

No significant difference in the score obtained by the subjects or time required to do the tasks was found between the two (viz and non-viz) groups. Thus results from first study were inconclusive. The second study was conducted after applying changes to the study and the material based on the problems found and users feedback from the first study. There was a significant difference among the scores of students using visualization versus non-visualization. The students in the visualization group performed well compared to those in the non-visualization group. These results suggest that visualization can be helpful in understanding and learning of the distributed algorithms.

The results obtained in this study are presented and discussed in more detail by another graduate student, Rong Wu, in her thesis (fall 2001).

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

Web personalization software: Two studies were performed. Both the studies used both undergraduate and graduate students:

Study one was designed to measure the use of the software in a given amount of time. The subjects performed two tasks, to develop either occurrence type or priority type applications. The results obtained from experiment one suggest that there was no significant difference between the scores of subjects developing occurrence type and priority type of applications. Further, no significant difference was found between the times required to build occurrence type and priority type application. Many changes were made to the software and the study based on the results and users feedback from the first study.

A second study was performed on the subjects to determine the usability as well as learnability of the web personalization software. There was no time limit for completing the task. The students were asked to perform only one task, to develop an occurrence type application. Around 70 percent of the students obtained full scores in the second study. However, in this study, the logic created by subjects was tested and they were asked to correct it if it was wrong. Checking the students results and asking them to correct their work is a significant contributing factor in their high scores. Feedback collected from this study suggests that the changes made to the software and the training material also helped students to perform well.

This study assisted in making the software better than what it was before the study. It also helped in evolving the training material, based on the results and the feedback from the study. The researchers believe that this study gives important feedback for further improvement of this software, so that it can be used ‘out of the box’ by the managers of the government firm. Once ready to be used formally by the managers it will help in increasing their work efficiency and productivity as well as also saving clients money and time.

Visualization: Two studies were conducted to see if visualization helps in learning and understanding of the software. The average scores obtained from the visualization and non-visualization group. There was no significant difference between the scores of the visualization and non-visualization groups in the first experiment for both the algorithms. The results and feedback from study two helped in recognizing the problems in the first study. The changes were made and the second experiment was conducted.

The second experiment shows that due to the changes and unlimited amount of time (*i.e.* they could stay for as long as they wanted, but were not forced to stay after 75 minutes) provided they performed really well as compared to the non-visualization group. Based on these results we can say that visualization can help in understanding of distributed algorithms. However it is very important that this study is repeated with more refined fashion on some more complex algorithms.

The first study was inconclusive. The results of second study indicate that visualization can be helpful in understanding the distributed algorithms used in this study. This is a big step in determining how visualization can help students in learning complex algorithms.

My contribution in the study of Web Personalization Software:

- Worked with the developer of the software and designed the study under supervision of Dr. Eileen Kraemer.
- Conducted the experiments.
- Collected and analyzed the data.
- Modified the tasks and the training material based on the users feedback, for the second study.
- Assisted the developer in modifying the software and fixing the bugs found during the first study.

My contribution in the Visualization Study:

- Participated in the discussion and meetings to develop the study.
- Implemented the demo visualization program (about finding the greatest value in an unordered array) used in the second. This was embedded in the same testing environment as the one used for the formal study.
- Provided the output and the pseudocode of the algorithms to be incorporated in the real study environment of the formal. This was used to make students familiar with the study environment.
- Helped in developing the tasks used in both study.

Both the studies conducted on the web personalization software determined the problems as well as useful features of the software. The results and the studies gave a fair evaluation of the learnability and the usability of the software. Chapters 3 and 4 and this chapter provide some more suggestions and recommendations for developing the software, so that it can be directly (without any prior training) used by the public managers. This study is a valuable step forward for this software to be used “out of the box” for the target users.

The following are specific recommendations so that the software can be used “out of the box”:

- This software is designed for public managers who may not be as computer literate as the subjects used in this study, therefore we recommend performing a usability study on a population with background more similar to the managers.
- Include some logical tables in examples, as shown in the directions for the second experiment, which cover all the possibilities to build logic for a particular condition.
- Directions supplied with the second study are recommended to be used with the software for future studies.
- Implementation of a “self testing feature” should be very helpful too.
- There are still some minor bugs in the web personalization software as described in chapter 4, which need to be fixed.

The visualization study helped in determining that visualization can be useful in understanding distributed algorithms. The implementation of the algorithms was one of the main elements of the study on which the visualization and the whole study was built. We found some positive results regarding the use of visualization, however, it would be interesting to see the effect of visualization on some more complex distributed algorithms. A comparative study with different visualizations on the same algorithm should be helpful, where we can evaluate different types of visualizations.

REFERENCES

1. Brown M. H., and Sedgewick R. 1985. Techniques for Algorithm Animation. *IEEE Software*, 2:1, 28-39.

2. Byrne, Michael D, Catrambone, Richard and Stasko, John, 1999. Evaluating Animations as student Aids in Learning Computer Algorithms. *Computer & Education*, 33:4, 253-278.

3. Card, S., Mackinlay, J., and Shneiderman, B. (Eds). Reading in Information Visualization: Using Vision to Think. Los Altos, CA: Morgan Kaufmann.

4. Carothers, C. D., Topol, B., Fujimoto, R. M., Stasko, J. T., and Sunderam, V. 1999. Visualizing Parallel Simulations That Execute in Network Computing Environments. *Future Generation Computer Systems*, 15, 513-529.

5. Dijkstra, E. W., and Scholten, C. S. 1980. Termination detection for diffusing computations. *Inf. Process. Lett.* 11, 1-4.

6. Dumas, J. S. and Redish, J. C. A Practical Guide to Usability Testing. Norwood, N.J.:Ablex, 1994.

7. Graham, M., Kennedy, J., and Benyon, D. 2000. Towards a methodology for developing visualizations. *Int. J. Human-Computer Studies* 53:789-807.

8. Hughes, M. 1999. Rigor in usability testing. *Technical Communication* 46(4):488-494.

9. Human-Computer Interaction. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, eds. Workingham, England:Addison-Wesley, 1994.

10. Kehoe, C., Stasko, J., and Taylor, A., 2001. Rethinking the Evaluation of Algorithm Animations as Learning Aids: An Observational Study. *Int. J. Human Computer Studies*, 54, 265-284.

11. Lecerof, A., and Paterno, F. 1998. Automatic support for usability evaluation. *IEEE Transactions on Software Engineering* 24:10, 863-887.

12. Lewis, C. Using the 'Thinking Aloud' Method in Cognitive Interface Design. Research Report RC9265, IBM Thomas J. Watson Research Center, Yorktown Heights, N. Y., 1982.

13. Mobasher, B. Cooley, R., and Srivastava, J. 2000. Automatic personalization based on web usage mining. *Communications of the ACM*, 43:8, 142-151.

14. Mulholand, P. 1998. A Principled approach to the evaluation of SV: A case study in Prolog. In J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, editors, *Software Visualization-Programming as a Multimedia Experience.* Chapter 29. MIT Press. Cambridge, MA. London, England. 439-452.

15. Mulvenna, M. D., Anand, S. S., and Buchner, A. G. 2000. Personalization on the net using web mining. *Communications of the ACM* 43(8):123-134.

16. Nielsen, J. *Usability Engineering*. Boston: Academic Press, 1993.

17. North, C., and Shneiderman, B. 2000. Snap-together visualization: can users construct and operate coordinate visualizations? *Int. J. Human-Computer Studies* 53:715-739.

18. O'Leary, Mick 1999. Web Personalization Does it Your Way. *Online,* 23:2, 79-82.

19. Randall, N. 1998. Making software easier through usability testing. *PC Magazine* 17(17):285-287.

20. Reed, S. 1992. Who defines usability? You do. *PC Computing* 5:12, 220-230.

21. Roman, G. C., and Cox, K. C. 1992. Program Visualization: The Art of Mapping Programs to Pictures. *Proceedings of the 14th International Conference on Software Engineering. Melbourne, Australia. May 1992*. 92:6, 412-420.

22. Rubin, J. 1994. *Handbook of Usability testing: How to plan, design, and conduct effective tests*. New York, NY: John Wiley and Sons, Inc.

23. Shneiderman, B. 1998. Designing the User Interface: Strategies for effective human-computer interation. Reading, MA:Addison-Wesley.

24. Smyth, B. and Cotter, P. 2000. A personalized television listings services. *Communications of the ACM*, Aug 2000; 43:8, 107-111.

25. Stasko, John, Domingue, John, Brown, Marc H. and Price, Blaine A. (editors), 1998. *Software Visualization: Programming as a Multimedia Experience*. MIT Press, Cambridge, MA.

26. Stasko, J. T., and Kraemer, E. 1993. A Methodology for Building Application-Specific Visualizations of Parallel Programs. *J. Parallel and Distributed Computing*, 18, 258-264.

27. Storey, M.A.D., K. Wong, H. A. Muller. 1997. How do program understanding tools affect how programmers understand programs? *Proceedings of the Fourth Working Conference on Reverse Engineering.* Amsterdam, Netherlands. 6-8.

28. Sutcliffe, A. G., Ennis, M., and Hu J. Evaluating the effectiveness of visual user interfaces for information retrieval. *Int. Human-Computer Studies* 53:741-763.

29. Topol, B., Stasko, J. T., and Sundram, V. 1998. PvaniM: A Tool for Visualization in Network Computing Environments. *Concurrency: Practice And Experience*, 10:14, 1197-1222.

30. Wells, N., Wolfers, J. and Riecken, R. C. 2000. Finance with a personalized touch. *Communications of the ACM*, Aug 2000. 43:8, 30-34.

31. Wharton, C., Rieman, J., Lewis, C., and Polson, P. "The Cognitive Walkthrough: A Practitioner's Guide," Usability Inspection Methods, J. Nielsen and R. L. Mack, eds. New York: John Wiley & Sons, 1994.

32. Zernik, D., Snir, M., and Malki, D. 1990. Using Visualization Tools To Understand Concurrency. *IEEE Software*, 9:3, 87-92

APPENDIX A

HANDOUT

Consent form:
The purpose of the study is to better understand the potential for governments to develop customized information and service delivery via the Internet. In particular, we want to know just how difficult and time-consuming it is likely to be for public managers to build customization software applications. As most public managers have education levels similar to yours, your performance on a task of software application building will give us some insight into the likely performance of public managers. The task will likely take most of the class period and will involve the use of logic and the ability to learn a new software interface.

During (or after) participation you could experience discomfort because of a feeling that your individual performance is being judged. You should be assured that this is not the case. While your individual performance time and score will be recorded, it will not be identified or connected in any way with you. To protect your confidentiality, we are asking participants to use a code word to identify their work and other information provided. As the researchers will not know this code word, your individual performance cannot be known to the researhers, your instructor or others. Hence, there is almost no risk that your performance could affect your class standing/grade or change anyone's perceptions of your ability. Also, while we greatly appreciate your help in this research effort, your participation is entirely voluntary. The researchers can be reached by telephone at: 706-542-5799 , 706-542-0597 and 706-542-9606 and by email at: eileen@cs.uga.edu , olooney@cviog.uga.edu and dhawan@cs.uga.edu.

No discomfort and stresses are foreseen.

No risks are foreseen.

I agree to take part in a research study titled "Customization/Personalization Software Usability Study", which is being conducted by Eileen Kraemer (Computer Science Dept., 706-540-5799), and her graduate student Ritu Dhawan under the direction of Eileen Kraemer and John O' Looney (Carl Vinson Institute of Government, 706-542-9606). I do not have to take part in this study; I can stop taking part at any time without giving any reason, and without penalty. I can ask to have information related to me returned to me, removed from the research records, or destroyed. I understand the procedures described above and I am over 18 year old.

My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

		/ /2001
Researcher's Name (Please Print)	Researcher's Signature	Date
		/ /2001
Participant's Name (Please Print)	Participant's Signature	Date

For questions or problems about your rights please call or write: Chris A. Joseph, Ph.D., Human Subjects Office, University of Georgia, 606A Boyd Graduate Studies Research Center, Athens, Georgia 30602-7411; Telephone (706) 542-6514; E-Mail Address IRB@uga.edu.

Questionnaire:
NOTE: Please take one copy of the consent form with you and give one to the researcher.

Id Code: ________________ (same as the userID you will use on the login).

Time to Start Sample Exercise:___________
Time of Finishing Sample Exercise:_________
Time to Start Exercise 1:___________ **Time of Finishing Exercies1**:_________.

Type of Task: (circle one) Occurrence Priority

General experience and comfort with learning to use new software applications without instruction:
Little 1 2 3 4 A great deal
5

How important do you think it is that governments provide web sites that have been personalized to the individual needs of citizens?
Very Unimportant 1 2 3 4 5 Very
Important

To what degree do you think that a college graduate in a non-technical field would be able to create a web site that was personalized for individual citizens?
Very unlikely 1 2 3 4 Very likely
5

Rate the degree to which the following were barriers to using the software:
Analyzing the exercise problem to create the logic:
Not a barrier 1 2 3 4 Very much a barrier
5
Using the software interface to implement the logic:
Not a barrier 1 2 3 4 Very much a barrier
5

Do you think you are capable of teaching others to use this application?
Very unlikely 1 2 3 4 Very likely
5

Given the nature of the task, to what degree were the directions clear and understandable? ***(If you rate as unclear, please specify or note on directions where this is the case)***
Very Unclear 1 2 3 4 5 Very Clear

Estimate (as a percent of the total time you spent on Exercise 1, the first exercise) the time you think it would take you to complete another assignment of similar difficulty.

Circle one: [100% 90% 80% 70% 60% 50% 40% 30% 20% 10%]

What kind of problems did you experience with this application?

What design feature did you find useful or helpful?

What suggestions do you have for improving the usability of this application?

Directions: Directions for Experiment # 1: Please refer to the following website for detailed directions and examples used to develop web personalization application.

http://webster.cs.uga.edu/~dhawan/personalization.html

Directions for Experiment # 2:

This experiment involves creation of personalized web pages using personalization software. In this experiment you will perform two exercises, attached at the end of this handout and entitled, (1) *Emergency Management Application Marketing Report* and (2) *Health and Wellness Advisor Application Marketing Report*. A step-by-step procedure and strategies to perform the tasks with related examples are also included in this handout.

The three main steps in creating a personalized application are:

1. **Creating profile question/statements.** These profile questions are used to classify users into groups and are the basis for the selection of messages to present.
2. **Adding messages.** These are the messages that will appear on the personalized web page.
3. **Building the logic**. This determines which messages each group of users will receive.

Before performing the real exercises you will perform a SAMPLE exercise to familiarize you with the real exercises. In case the description provided in the handout is not sufficient, a detailed description and related examples are also available at: http://iep.cviog.uga.edu/ptechnew/directions.asp

The SAMPLE exercise will help you to understand the steps involved in completing the real exercises and to get the feel of the final product**. Go through the sample exercise on page A1 very carefully and follow the directions given below**. After you finish the SAMPLE exercise you will follow the same steps to complete the two exercises provided at the end of the handout.

Directions

1. Choose the type of application you want to develop, ***Occurrence*** in your case.
2. Type in a unique ID for yourself to login and click on the "Begin Application Building" button.
3. Record this ID and start and stop time of your sample test, Exercise 1 and Exercise 2.
4. Directions are provided on each page we will go through to build the application. However, you should focus on the directions provided on this handout and refer to the onscreen directions only if you need further assistance.

Step 1: Create New Profile Set

Once you have logged in, you will be taken to a Control Page where you will begin to build an application. The type of application you chose will be displayed. Since you are using this software for the first time, choose the ***New Application*** button

- Click on the "**New Application**" button to go to the "**Personalization System Creation Form"**. When you reach this form, you will notice that it has already been given a unique name and that two sample profile questions have been filled in.
- Click the submit button to finish setting up an application that provides personalized messages to people based on their level of physical fitness and whether they smoke or not.

  NOTE: you should only use **YES-NO** (this choice automatically provides a Yes and No label for these two answer choices) or **SCALE** (this choice automatically create a response set of five choices, essentially 1 to 5 on a 5 point Very False-to-Very True Scale)**.** You can ignore Weight and other optional factors that are marked in blue.

- After submitting the Creation Form, you will see two buttons.

  Step 2: Add Messages and **Go to Control Panel**

Step 2: Add Messages

By clicking the **Step 2: Add Messages** button you will be taken to a form where you can begin creating messages:

NOTE: For this sample exercise you need a total of two messages, provided below:

- On this page you will see a text field labeled as message #1.

- Type in the following message in that field:

"You have a high risk of heart attack"

- Click on the "**Add Message**" button.
- Next you will be given the options to correct your message, add another message, or move on to Step 3 (building Logic).
- Click on the "**Add Another Message**" button and you will see the label of the message text field changed to "message # 2".
- Enter the following message in the message # 2 field.
  "You have a low risk of heart attack"
- Click the "**Add Message**" button.
- Click on "**Finish adding messages Go to step3: Building logic**"

Step 3: Building Logic

This will bring you to an "**Adding Logic"** form where you will begin to create a logical link between the Profile statements/questions set in Step 1 and the Messages you created in Step 2.

I exercise frequently.

Verv false Verv true

Are you a heavy smoker?

	1	2	3	4	5
Yes	High Risk	High Risk	High Risk	High Risk	High Risk
No	High Risk	High Risk	High Risk	Low Risk	Low Risk

Here you specify the required pattern by using the series of condition **Operator** radio buttons in combination with the series of **Value** radio buttons. The Operator buttons are: = [Equal to], LT= [Less Than or Equal to], GT= [Greater Than or Equal to].

Value buttons include an **NA** or Not Applicable button and five more buttons that have underlying values ranging left to right from 1 to 5. (Note: if you specified a Yes/NO, the value for Yes is 1 and value for No is 2). Using a combination of an Operator and a Value button, you can specify that a text will be returned based on any set of values (both within a single condition statement or across a number of condition statements). The default Value button begins with each condition statement being NA. You will need to specify at least one of the statements as applicable to obtain an answer.

The chart above indicates the profile associated with low or high risk of heart attack. Three logic statements, corresponding to the three marked boxes, are necessary to specify this.

Logic 1: People who answer “Yes” for condition 1 (Are you a heavy smoker?) are at high risk, no matter how they answer condition 2 (I exercise frequently). This can be indicated by selecting greater than or equal to 1 for condition 2 or by selecting N/A for condition 2.

Logic 2: People who answer “No” for condition 1 (Are you a heavy smoker?), and less than or equal to 3 for condition 2 (I exercise frequently) are at high risk.

Logic 3: People who choose “No” for condition 1(Are you a heavy smoker?) and greater than or equal to 4 for condition 2 (I exercise frequently) are at low risk.

NOTE: you have to always choose “=” for operator button if it’s a YES/NO type question.

Important Tip: If you want a text block to be returned no matter what, you will need to set one of the statements to a logic value of 1 (or very false) and keep the default direction value of Greater than or equal to (GT=).

These logics can be implemented using the following steps:

- Choose from **“This Message #:…..”** window, a message you want to be returned based on the logical link you will create.
- Choose a combination of operator and value button based on the above example of logic to create a logical connection.

- After creating each logic click on the “Make Logical Connection“ button.
- You will get a message box. Confirm by clicking “OK” on the text message box.
- This will lead you to a page where you have option to, **Make More Logical Connections**, **Review your Messages/Logic**, **GO to Step 4 (Finish & Test the Application) or Go to the Control Panel**.
- Click on “Make More Logical Connections” and repeat the same steps given above till you have made at least one logical connection with each messages. (In this case you have to create 3 logics for two messages to achieve logic shown in the chart above).

NOTE: Besides these examples you can make more message/logic connections for each message.

- If you want to revise your logical connection, click on “**Review your logic and messages**” and click “**review link**” besides the message you want to revise the logical link of. You can also add more logic sets to the message by clicking on “**Add link**” besides the message you want to add more logical link to.

- After making all logical connections we can click on button “**GO to Step 4 (Finish & Test the Application)**” to test the application we just built.

Step 4. Test the correctness of your logic

- If you choose the “**Finish and Test**.” button, you will then see a form that replicates what a potential client or user might see. Click on the various radio buttons, to provide profile data, and click the “**Test Application**” button.
- Test the various responses to the condition statements to be sure that they return the appropriate text.

Example: In this sample test:
If you choose:
Are you a heavy smoker? As Yes, and
I exercise frequently. As (=LT) 3
and then click on “**Test Application**”. You should be able to see “You have a high risk of heart attack” or
If you choose:
Are you a heavy smoker? As No, and
I exercise frequently. As (=GT) 4
and then click on “**Test Application**”. You should be able to see “You have a low risk of heart attack”
If you do not get the correct response, you can go back to the “**Control Page**” and then click the “**Review Messages/Logic**” button to revise your logic.

5. **You can always go to the Control Panel and revise the application you have built.**
6. **The material following this consists of the strategies and some examples for performing the exercises and the two exercises you have to complete in this experiment. You will create a total of two applications, one for each exercise. Name your applications exactly as specified below and then follow the same step-by-step directions you used to create the sample exercise to complete each of them. In both the exercises you need to create 3 condition question/condition statements, 6 messages and 10 logic sets. For each application you will go to the control page and start by clicking on the “**New Application**” button. This will take you to the** Create application form **page; give a name to your application as specified below.**
   Name for exercise 1: Emergency + your ID
   Name for exercise 2: Health + your ID

Remember to record start and stop times on the questionnaire.

Steps to perform Exercise 1: Emergency Management Application Marketing Report

Please go through Exercise 1(pages B1 - B3) very carefully and then perform the following steps.

1. Go to the control page to start the new application.
2. Click on create/load another application.
3. Click on new application.
4. Fill in the new name for your application, *Emergency* + your ID in this case.
5. Erase the profile questions already filled in and type in the following questions in the profile statement/question box and specify the kind for each in the drop down "Kind Menu". These conditions are based on conditions given in exercise 1, under Issue A, B and C.

NOTE: There are three different issues in exercise 1 and all of the can be addressed asking following questions:

	Statement/questions	**Kind**
1.	Your family has many children.	SCALE
2.	Do you have electronic gear?	YES/NO
3.	You own an aggressive pet.	SCALE

6. After filling the form, click on submit.
7. Click on "Add Messages".
8. Add messages, a total of **6 in this case**. One specifying high risk and another low risk for each emergency conditions given in the exercise and as we did for the sample test. Follow the exact directions in **step 2** mentioned above.

NOTE: Keep your messages very simple: "You are at [high/low] risk of .....". Each message should convey only ONE idea (e.g., "You are at high risk of illness", NOT "You are at high risk of illness and high risk of crime.").

9. Create logic links, a total of **10 logical connections** for this exercise. You have to create logic depending conditions given in each issues in the exercise. The number of logical pattern you have to create for each issue and type of issue (one condition, two condition or three condition are given in the exercise). One message (e.g., "You are at high risk of illness") will sometimes need to be linked to more than one logical pattern depending on the conditions specified in the exercise**. This is the main part of the exercise, and it is very important for you to correctly specify the logic patterns for each type of issue. If you wish you can use the scratch paper provided to create charts as shown in the sample test to build your logic.**

NOTE: Examples to create logic for one condition factor and two conditions factor are given in the strategies to perform the application, pages 10 - 14.

9. Test the logic for correctness.

Steps to perform Exercise 2: Health and Wellness Advisor Application Marketing Report

1. Exercise 2 is described in page C1.

2. Follow the same procedure as described for exercise1.

3. Name your exercise 2 as *Health* + your ID.

EXERCISE 1: Emergency Management Application Marketing Report

Your goal in personalizing the emergency management section of the web site should be to provide the citizen-users with a sense of their risk (HIGH or LOW) for each of the following dangers:

A. Sudden illness
B. Sudden Illness
C. Neighborhood Dispute Risk
D. Crime

To keep the messages simple, you will need only to indicate whether a family is likely to be at high or low risk for each of these threats. For the **entire exercise** you will need to produce:

- ***3 condition statements/questions***
- ***6 messages (a high and a low risk message for each area)***
- ***10 logic patterns/links***

Issue A. *Illness Risk* [2 logic patterns]
One Condition/profile factor affects illness risk:

- All else being equal, the more children in the family the more the family is at risk of a sudden illness. (SCALE)

Issue B. Neighborhood Dispute Risk [3 logic patterns]

Two Condition/profile factors affect dispute risk:
- *No matter how other factors affect risk for neighborhood disputes*, families with electronic gear are at high risk for neighborhood disputes. (YES/NO)

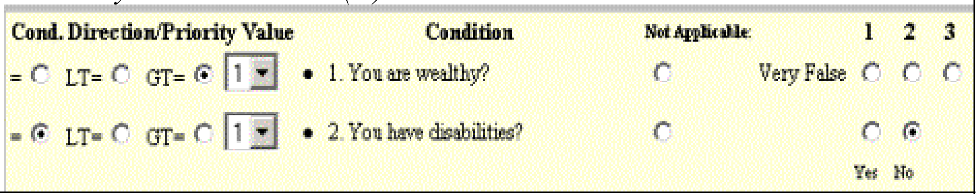
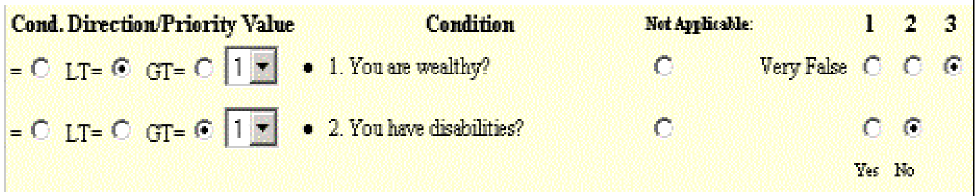
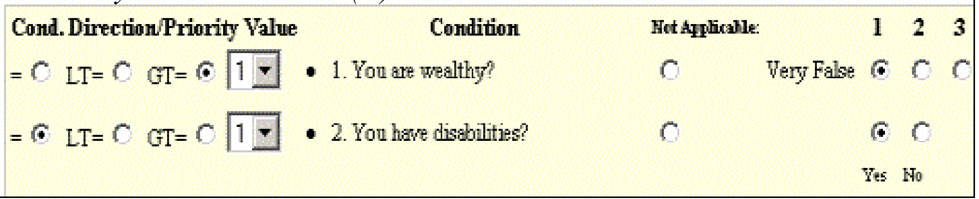
- All else being equal, the more aggressive the family pet, the more likely the family will be at higher risk of a neighborhood dispute. (SCALE)

GUIDING EXAMPLE for Dealing with Issue B

All else being equal, wealthier people are more likely to need a reminder to file a luxury tax.

No matter how other factors affect the likely need to file a luxury tax, a person with a disability will be at a low risk of having to file a luxury tax.

Example Text-Logic Patterns Needed

Text Message	***Logic***
1. HIGH RISK: You are very likely to need to file a luxury tax.	*1. Wealth condition is equal to or greater than (=GT) 4.* *Disability condition is No (=) NO.*  Cond. Direction/Priority Value — Condition — Not Applicable: — 1 2 3 = ○ LT= ○ [✓ GT=] 1 • 1. You are wealthy? ○ Very False ○ ○ ○ [✓ =] LT= ○ GT= ○ 1 • 2. You have disabilities? ○ Yes ○ [✓ No]
2.LOW RISK: You are not likely to need to file a luxury tax.	*2. Wealth condition is equal to or less than3. (=LT) 3* *Disability condition is No (=) NO.*  Cond. Direction/Priority Value — Condition — Not Applicable: — 1 2 3 = ○ [✓ LT=] GT= ○ 1 • 1. You are wealthy? ○ Very False ○ ○ [✓ 3] = ○ LT= ○ [✓ GT=] 1 • 2. You have disabilities? ○ Yes ○ [✓ No] *3. Wealth condition is equal to or greater than (=GT) 1. (I.e., any wealth condition)* *Disability condition is Yes (=) Yes.*  Cond. Direction/Priority Value — Condition — Not Applicable: — 1 2 3 = ○ LT= ○ [✓ GT=] 1 • 1. You are wealthy? ○ Very False [✓ 1] ○ ○ [✓ =] LT= ○ GT= ○ 1 • 2. You have disabilities? ○ [✓ Yes] No ○

Issue C. Crime Risk [5 logic patterns]

Two Condition/profile factors affect crime risk:

- All else being equal, families with more aggressive pets are at higher risk of crime. (SCALE)
- All else being equal, families with more children are at higher risk of crime. (SCALE)

While a number of different logic patterns could be used to establish high and low risk, for the purpose of this exercise, you are instructed to provide a **high risk message** when: at least one condition is at the True or Very True level (i.e., equal to or greater than 4), while the other condition is at least at the neutral level (i.e., equal to or greater than 3). It takes two logic patterns to state this. **Because the low risk** group must include all of the remaining possibilities, this would mean that if either condition were less than the neutral level (i.e., equal to or less than 2), the person would be of low risk –no matter what the value of the other condition was. Low risk would also occur when both conditions are neutral. It takes three logic patterns to state this.

Two-Factor-Plus-Qualification Scenario

All else being equal, wealthier people are more likely to need a reminder to file a luxury tax. (SCALE)

All else being equal, the more homes people own the more likely they are to need to file a luxury tax. (SCALE)

Example Text-Logic Patterns Needed

Text Message	***Logic Settings***
1. HIGH RISK: You are very likely to need to file a luxury tax.	1. Wealth condition is equal to or greater than (=GT) 4. Home ownership condition is equal to or greater than (=GT) 3. 2. Wealth condition is equal to or greater than (=GT) 3. Home ownership condition is equal to or greater than (=GT) 4.
2. LOW RISK: You are unlikely to have to file a luxury tax.	3. Wealth condition is equal to or greater than (=GT) 1. Home ownership condition is equal to or less than (=LT) 2. 4. Wealth condition is equal to or less than (=LT) 2. Home ownership condition is equal to or greater than (=GT) 1. 5. Wealth condition is equal to (=) 3. Home ownership condition is equal to (=) 3.

EXERCISE 2: Health and Wellness Advisor Application Marketing Report

Your immediate goal in personalizing the health and wellness section of the web site is to provide the citizen-users with a sense of their level of risk (HIGH or LOW) for each of the following dangers:

A. Heart attack
B. Bone disease
C. Cancer

Keep the messages simple: only indicate whether a family is likely to be at high or low risk for each of these threats

For the **entire exercise** you will need to produce:

- ***3 condition statements/questions***
- ***6 messages (a high and a low risk message for each area)***
- ***10 logic patterns/links***

Our research indicates that:

Issue A. Heart Attack [2 logic patterns]

- All else being equal, the older one is the higher one's risk of a heart attack. (SCALE)

Issue B. Cancer [3 logic patterns]

- All else being equal, the more one exercises, the less risk of cancer. (SCALE)
- No matter how other factors affect risk for cancer, if one smokes, one is at high risk of cancer. (YES/NO)

Issue C. Workplace Injury [5 logic patterns]

- All else being equal, the older one is the higher risk for workplace injury. (SCALE)

- All else being equal, the more one exercises, the higher the risk of workplace injury. (SCALE).

While a number of different logic patterns could be used to establish high and low risk, for the purpose of this exercise, you are instructed to provide a **high risk message** when: at least one condition is at the True or Very True level (i.e., equal to or greater than 4), while the other condition is at least at the neutral level (i.e., equal to or greater than 3). **Because the low risk** group must include all of the remaining possibilities, this would mean that if either condition were less than the neutral level (i.e., equal to or less than 2), the person would be of low risk –no matter what the value of the other condition was. Low risk would also occur when both conditions are neutral. It takes three logic patterns to state this. (see example from first marketing report).

SAMPLE EXERCISE: Sample Application Marketing Report

Application is already given a *unique* name

Your goal in personalizing the sample web site should be to provide the citizen-users with a sense of their risk (HIGH or LOW) for the following danger:

A. Heart Attack
To keep the messages simple, you will need only to indicate whether a family/person is likely to be at high or low risk for each of these threats. For the **entire exercise** you will need to produce:

- ***2 condition statements/questions*** (Already created and filled in for sample test)
- ***2 messages (a high and a low risk message for each area)***
- ***3 logic pattern/links***

NOTE: For the sample exercise you will be provided with the statement/questions and messages.

Issue A. Heart Attack

Two Condition/profile factors affect heart attack risk:

- *No matter how other factors affect risk for heart attack,* a person who smokes is at high risk of heart attack. (YES/NO)

- All else being equal, the less the person exercises, the higher is the risk of heart attack. (SCALE)

Statement/question: These questions/statements are based on the conditions provided above.

1. Are you a heavy smoker?
2. I exercise frequently.

NOTE: These statement/questions are already provided and typed in for you. In the real exercises you will provide your own statements and questions.

Message: These messages are based on the danger (Heart attack) provided above.

1. You have a high risk of heart attack.
2. You have a low risk of heart attack.

APPENDIX B

RAW DATA

Raw Data For Experiment # 1 on Web Personalization Software (occurrence type application):

Code	Correctness		Exercise 1			Exercise 2		
	Ex1	Ex2	start time	stop time	Diff	start time	stop time	Diff
smith	6	5	3:45	4:00	15	4:01	4:08	7
bob	1	7	12:35	1:04	29	1:06	1:31	25
666	0	0	2:30	3:10	40	3:10	3:30	20
jcc	0	0	4:50	5:15	25	5:15	5:30	15
Rugger	1	0	4:50	5:25	35	5:25	5:45	20
conin666	0	0	4:55	5:30	35	5:30	5:50	20
Belllover	6	5	6:45	7:15	30	7:15	7:43	28
cs6800	1	2	12:33	1:40	67	1:40	1:45	5
quirk	0	0	2:40	3:00	20	3:00	3:15	15
sisyphus	1	1	4:55	5:45	50	5:45	6:05	20
mitchell	5	5	3:55	4:11	16	4:13	4:18	5
mi	8	9	3:00	3:17	17	3:17	3:32	15
vicki	6	5	10:15	10:45	30	10:45	11:03	18
crazy	5	7	10:15	10:55	40	10:55	11:10	15
smlee	3	4	2:46	3:05	19	3:05	3:13	8
BEP	7	0	11:14	11:46	32	11:47	12:04	17
Jinling	1	2	11:30	11:50	20	11:50	12:15	25
sahib!	7	9	1:00	1:12	12	1:12	1:24	12
paperclip	8	0	12:40	1:30	50	-	-	
reux	0	0	1:10	1:16	6	1:16	1:24	8
CME	5	5	7:13	7:34	21	7:35	-	
mean	3.38	3.14			29			15.68
var	9.14	10.12			216			46.89
std	3.02	3.18			14.69			6.84

Raw Data For Experiment # 1 on Web Personalization Software (priority type application):

code	Correctness		Exercise 1				Exercise2	
	Ex1	Ex2	start Time	stop time	time diff	start time	stop tme	Time diff
shah4356	10	10	3:45	4:00	15	4:00	4:06	6
Superman	5	5	6:53	7:32	39	7:32	7:54	22
tata	0	0	10:22	10:30	8	10:31	10:39	8
Holly	2	2	5:33	5:45	12	5:46	5:53	7
toons	5	7	12:50	1:10	20	1:10	1:26	16
jayjay	6	0	1:06	1:19	13	1:20	1:24	4
br-30605	0	0	12:47	1:05	18	1:06	1:23	17
alpha1	10	10	12:57	1:08	11	1:08	1:17	9
cowboy	0	0	12:47	1:12	25	1:13	1:35	22
Q26789	0	0	12:35	1:30	55	1:30	1:45	15
IS8526	5	0	12:30	1:40	70	1:41	1:50	9
nebraska	6	4	2:40	3:15	35	3:15	3:30	15
newbie	8	5	2:37	2:52	15	2:52	3:04	12
hahaha	2	2	4:53	5:25	32	5:25	5:52	27
mean	4.21	3.21			27.15			14.07
var	13.10	13.72			348.47			45.91
std	3.61	3.70			18.66			6.77

APPENDIX C

ALGORITHMS

1) Dijkstra-Scholten Distributed Diffusing Termination Detection

```
//Node Class: each node contains pipes to read incoming messages and send
//outgoing messages. contains parent in not root.
//starts the thread if the node is a root.

public class Node{
   int parent = -1; //parent ID or -1 for no parent
   int n= 0; //number of nodes
   public  Pipe[] outgoing; //outgoing link to node i; null for no link
   public Pipe[] incoming;  //incoming link from node i, null for no link
   public boolean root = false; //true if this is a root
   public int myid = 0; //the ID of this node
   boolean idle = true; //true if this node is idle (not working)
   int count = 0;  //counter to keep track of messages not acknowledged

   public Node(int num, int id){   //initialize the node
         n = num;
         myid = id;

         //at most n outgoing and incoming links
         outgoing = new Pipe[n];
         incoming = new Pipe[n];

         //initially this node is not connected to any other node
         for(int i=0; i<n; i++){
                 outgoing[i] = null;
                 incoming[i] = null;
         }
   }//end of Constructor

   public void addOutLink(Pipe p, int neigh){//add outgoing link to node neigh
         System.out.println("NODE "+ myid+": outgoing link to "+ neigh);
         outgoing[neigh] = p;
   }//end of addOutLink

   public void addInLink(Pipe p, int neigh){ //add incomming link from node neigh
         incoming[neigh] = p;
         System.out.println("NODE "+ myid+": incoming link from "+ neigh);
   }//end of addInLink
   public void run(){ //what this node does
         int aux;
         System.out.println("NODE "+myid+": "+"root is "+root);   //write the current
values of the variables
         System.out.println("NODE "+myid+": "+"counter is "+counter);
         System.out.println("NODE "+myid+": "+"idle is "+idle);
         System.out.println("NODE "+myid+": "+"parent is "+parent);
         if(root == true)
            Application.sendMsg();  //the underlying application may send messages to
other nodes
         while(true){              //this is a daemon, so it runs continuously
            for(int i=0; i<n; i++){  //check each possible link of the node
                  if(incoming[i] != null){    //there is a link from i
                     aux=incoming[i].get();   //get the message if any
                     if(aux != NO_MSG){      //if there is a message
                           System.out.println("NODE "+myid+": message type "+aux+" from
node "+i);
                           if( aux == APP_MSG){ //application message
                              idle = false;   //start working
                              System.out.println("NODE "+myid+" :"+ "idle is"+ idle);
                              if(parent==-1 && root==false){
                                    parent = i;
                                    System.out.println("NODE " + myid+": parent is "+
parent); }
                              else{  outgoing[i].put(ACK_MSG);   //send an
acknowledgement
                                       System.out.println(" NODE " + (char)('A'+myid)+
                                                ": Sending Ack to the parent "+(char)('A'+Parent));
}
                              //the underlying application may send messages to other nodes
                              Application.sendMsg();
                           System.out.println("NODE "+myid+": App message sent to
                           "+otherNodes);
                             System.out.println("NODE "+myid+": counter is "+counter);
                  }
                           if(aux == Pipe.Ack){ //message is an acknowledgement
                              count--;
                              System.out.println("NODE "+myid+": counter is "+count);  }
                     }//if (aux != NO_MSG)
                  }//if (incoming[i] != null)
            }//end of for loop

            Application.checkIdle();  //check if the underlying application finished; this
might modify idle
            System.out.println("NODE "+myid+": done is "+done);
```

```
  if(root && idle == true && count ==0)
                System.out.println("Node "+myid+": task finished");
           if(!root && parent != -1 && idle == true && count==0){  //acknowledge to
parent
                outgoing[parent].put(ACK_MSG);
                System.out.println("NODE "+myid+ ": "+ "ack sent to parent "+parent);
                parent = -1;
                System.out.println("NODE "+myid+ ": "+ "parent is "+parent);   }
        }//end of while(true)
   }//end of run method
}//end of class Node
```

Explanation of Dijkstra-Scholten Distributed Diffusing Termination Detection

The algorithm presented here assumes a distributed environment, which consist of a set of nodes that can communicate with each other only through message passing. The nodes are independent machines, like the ones that form the Internet, and they are linked to some of the other nodes (not necessarily all) by links or channels. A node knows only the number and the name of its neighbors, i.e., nodes to which it has a link. The only interaction between the nodes is via messages sent or received on the links. For example, it is impossible for a node to find the value of a variable A at one of the neighbors directly; it must send a message to ask for that value, and if the neighbor understands the request, it sends the value back.

A termination detection algorithm is designed to allow a node to determine if a task that is computed by multiple nodes has finished or not. Such a task (also referred to as an application) can be a project on which multiple people, each at home and using only email, work. The project might be a software product, which is divided among multiple people. Each person knows how to do certain things and cannot perform another. After they program some parts of a module (i.e. the networking functions), they might need to send that module to some other people so they can work out other aspects of the module (such as the dialog interface). At any time they might receive a request to do something from the people they know of (neighbors in the distributed environment). They can decide to work on that request, or to divide it among other people, or both.

The problem is how the initiator of the project detects the project's completion. It is possible that people not known to the initiator are still working on the project. Furthermore, even if everybody responds that they are done, the assumption that the task is finished might not be correct. A request might be in transit between two people that are not working, and neither the sender or the receiver might be aware of the status of the request: the sender might assume that the request was already processed, and the receiver might not know that a work request is going to arrive. Thus, the task is not finished until the job in the request is performed.

The termination detection algorithm is design to provide the initiator of the project with an answer of whether the application (project) is finished. Note that the project itself can be any distributed application (any algorithm) not only the example presented above. Termination detection works on top of this underlying application, i.e., people are still performing their duties, but they also do the termination detection.

The termination detection needs to know three things about each node (person), whether they are idle or working, the number of other nodes to which this node has sent a request, and the person that gave them a request. Three variables are kept at each node (and because this is a distributed environment are known only to that node):

-a boolean named *idle*, which is false when the node is doing something.

-an integer named *count* which contains the number of application messages (requests) that have not yet been acknowledged. An application message is any message that would be sent by the application even when the termination detection is not used. An acknowledgement is a new type of message different from any application message.

-a string named *parent* that contains the name of the node that gave some work to this node.

The *parent* is initially *null*, to show the fact that the node is not working on a task. The initiator (called root) also does not have a parent. When a node receives a request (application message) and it has not already had a parent, then its *parent* becomes the node that sent the request. A node, however, does not change its *parent* if it is not null.

When a node is done, it sends an acknowledgement to its parent. The node replies immediately with an acknowledgement to every application message that was received after its *parent* was set.

The *parent* variable creates a virtual tree of all the nodes that are involved in computing the task (work for the underlying application). This tree can expand as more nodes are involved in the application and can shrink as they finish their parts, due to the distributed nature, it is possible for a part of the tree to shrink while another part expands. A node can enter and exit the tree multiple times for the same project.

A node exits the tree only when it knows it is done (although it may be activated again later). For a node to be done, it is not enough to be idle, it also needs to have no other children. Otherwise, those children might be still working on the project, and the detection might give false answers.

The project is done when the root is done. This property can be detected by the root and ensures that no node is working on the project.

Question instructions:

-assume that at most one application task is performed in the entire network at one time

2) Distance Vector (Bellman-Ford) Routing Algorithm

```
//Node Class: each node contains pipes for incoming and outgoing links.

//each node is a separate process
public class Node extends Thread{
   //true if the table has changed since the last update
   boolean changed = false;
   int n= 0;    //the max number of nodes in the network
   public RoutingTable table;  //routing table
   public  Pipe[] outgoing;  //outgoing channels
   public Pipe[] incoming;   //incoming channels
   public int myid;  //this node's id

   public Node(int num, int id){
         n = num;
         myid = id;

         //reset the links
         outgoing = new Pipe[n];
         incoming = new Pipe[n];
         for(int i=0; i<n; i++){
            outgoing[i] = null;
            incoming[i] = null;
         }
   }//end constructor

   //method to add outgoing link
   public void addOutLink(Pipe p, int neigh){
         outgoing[neigh] = p;
         System.out.println("Node "+myid+" outgoing link to "+neigh);
   }//end of addOutLink

   //method to add incomming link
   public void addInLink(Pipe p, int neigh){
         incoming[neigh] = p;
         System.out.println("Node "+myid+" incoming link from "+neigh);
   }//end of addInLink


   public void run(){

         //updates to neighbors are always going
         //to contain the self destination at distance 0
```

```
//the destination table contains the self destination, with
//the next hop the node itself, at distance 0
table.addEntry(myid, myid, 0);

int l=0;
 //check each channel of that node
for(int i=0; i<n; i++){
   if(incoming[i] != null){
         //add an entry to the table for each neighbor
         table.addEntry(i, i, 1);
   }//end of if
}//end of for

System.out.println("Node "+myid+": "+table);

//this broacasts the destinations and their corresponding
//distances to all neighbors
broadcastTable();

while(true){//always look for update
   for(int x=0; x<n; x++){
         if(incoming[x]!=null){ //connected to x
            if(incoming[x].recovered){ //x just came up
                  //add it to the table
                  table.addEntry(x, x, 1);
                  changed=true;
            }
            if(incoming[x].failed){ //if link failed
                  //invalidate all table entries that have x as
                  //the next hop
                  invalidateNextHop(x);

                  //make sure the new table gets broadcast
                  changed=true;
            }
            else if(incoming[x].hasMsgs()){//a table was received
                  comTable = incoming[x].get();
                  System.out.println("Node "+myid+": received from "+x+
                        " update "+printTable(comTable));
                  for(dest in comTable){
                     //get the entry for "dest" in the table
                     //an entry is a 3-uple <dest, next, distance>
                     TableEntry e=table.getEntryFor(dest);
                     if(e==null){//don't know about dest
                           table.addEntry(dest, x,
                              1 + comTable.distanceTo(dest));
                           System.out.println ("Node "+myid+": dest "+
                               dest+", next hop "+x+
                               ", distance "
                               +table.distanceTo(dest));

                           changed = true;
                     }//end of if

                     else if(comTable.distanceTo(dest)+1 <
                               table.distanceTo(dest)){

                           //record a shorter distance to an
                           //existing destination
                           table.modifyEntry(dest, x,
                               comTable.distanceTo(dest)+1);

                           System.out.println("Node "+myid+": dest "+ dest+
                              ", next hop "+x+", distance "+table.distanceTo(dest));
                           changed = true;
                     }//end of else if
                  }//end of for all dest

                  for(dest in table){
                     //an entry is a 3-uple <dest, next, distance>
                     TableEntry  e=table.getEntryFor(dest);

                     //if x is used to reach e.dest
                     if(e.next==x){
                           //an entry is 2-uple <dest, distance>
                           UpdateEntry upEnt=comTable.getEntryFor(e.dest);
                           if(upEnt != null &&
                              e.distance != upEnt.distance+1){
                               //need to change this distance
                               table.modifyEntry(e.dest, e.next,
                                               upEnt.distance+1);

                              System.out.println("Node "+myid+": dest "+dest+
                             ", next hop "+x+", distance "+table.distanceTo(dest));
                             changed=true
                           }
                           if(upEnt == null){
                               //x does not know about e.dest anymore
                               table.removeEntry(e);

                           System.out.println("Node "+myid+ ": invalidating dest
                           "+e.dest);
```

```
                                changed=true;
                            }
                        }//e.next==x

                        //if x is not the next neighbor
                        if(e.next!=x){

                            if(comTable.getEntryFor(e.dest) == null){
                                //x does not know about e.dest

                                //send this to x
                                changed=true;
                            }
                        }//e.next != x
                    }//for
                }//end of elseif
            }//if incoming[x] != null

            //if necessary broadcast updates
            if(changed){
                System.out.println("Node "+myid+": table "+table);
                broadcastTable();
                changed=false;
            }
        }//for x=...
    }//while
 }//end of run method
 //invalidate all entries that have x as next hop
 //and prints out the invalidated entries
 public void invalidateNextHop(int x);


 //sends an update message to all neighbors
 //prints the neighbors that are included in the broadcast
 public void broadcastTable();
}//end of node
```

Distance Vector (Bellman-Ford) Routing Algorithm

The algorithm presented here assumes a distributed environment, which consist of a set of nodes that can communicate with each other only through message passing. The nodes are independent machines, like the ones that form the Internet, and they are linked to some of the other nodes (not necessarily all) by links or channels. A node knows only the number and the name of its neighbors, i.e., nodes to which it has a link. The only interaction between the nodes is via messages sent or received on the links. For example, it is impossible for a node to find the value of a variable A at one of the neighbors directly; it must send a message to ask for that value, and if the neighbor understands the request, it sends the value back.

A routing algorithm is designed to allow a node to address messages to any reachable node in the network. Because initially only the neighbors are known, it is impossible for a node to send a message to any other nodes except its neighbors. The routing algorithm assumes that nodes cooperate and forward each other's messages. In this way, a node can communicate to any other reachable node.

A routing algorithm must handle crashes and additions of both links and nodes. If a link or node fails, it may be necessary to use a different route between the same two nodes. Similarly, additional links or nodes might offer shorter routes or might make it possible to reach new nodes.

In distance vector routing a node knows, based on the *destination* of the message, to which neighbor to forward that message (*next hop*). Therefore, a node keeps a table that has three fields: *destination*, *next hop*, and *distance*. The first two fields are used for forwarding messages, while the third is employed to choose the shortest path. In most cases the shortest path excludes loops.

To build the routing table, neighbors tell each other what are the destinations they have heard of and what is the distance to that destination. When an update message is received (from one of the neighbors), a node checks to see if new destinations are included or if a shorter route exists to an already known destination.

Initially, a node knows only of its neighbors and starts with a table that contains one entry for each neighbor, with that neighbor as *destination*, as *next hop*, and with the *distance* equals to one. That implies that to send a message to a neighbor, the *next hop* is the neighbor itself at *distance* of one.

The table is modified only when an update message with new destinations or shorter distances is received, when a failure is detected, or when a new link appears (for simplicity, we don't discuss how a link failure or addition is sensed). In case of a link failure, all destinations that use that link are removed from the table. Note that a node crash appears to all its neighbors as a link failure; it is impossible for the neighbor to determine whether the link or the node has failed.

An update message is sent to all the neighbors in three cases:

-when the table is changed.
-when a new link is detected.
-when an update message from a neighbor does not contain a destination known to this node (The can happen after one of the neighbor's links has failed), or it contains a longer route to a destination.

Question instructions:

-stable state means that no update message is in transit on any link, and that the routing algorithm will produce no further update message before at least one link fails or appears. Messages that are being forwarded among nodes are not routing messages.