**RC23800 (W0511-110) November 14, 2005**
**Computer Science**

**IBM Research Report**

# An Evaluation of Using Programming by Demonstration and Guided Walkthrough Techniques for Authoring and Utilizing Deocumentation

**Vittorio Castelli, Madhu Prabaker*, Lawrence Bergman**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*Carnegie Mellon University

# An Evaluation of Using Programming by Demonstration and Guided Walkthrough Techniques for Authoring and Utilizing Documentation

## ABSTRACT

Much existing documentation is informal, and serves to communicate "how-to" knowledge among restricted working groups. Using current practices, such documentation is both difficult to maintain, and difficult to use properly.

In this paper, we propose a documentation system, called DocWizards, that uses programming by demonstration to support low-cost authoring, and guided walkthrough techniques to improve document usability.

We report a comparative study between the use of DocWizards and of traditional techniques for authoring and following documentation. The study participants showed significant gains in efficiency and reduction in error rates when using DocWizards. In addition, they expressed a clear preference for using the DocWizards tool both for authoring and for following documentation.

### Author Keywords

Programming-by-demonstration, Guided-walkthrough, Documentation authoring

### ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces. – Training, help, and documentation

## INTRODUCTION

Today's computer users are frequently confronted with navigating complex applications to perform tasks. Examples range from configuring connectivity settings to accommodate a new ISP, to performing a complicated formatting task within a word processor, to installing multi-application software systems, to setting up software environments for specific projects. Guidance in performing such tasks is obtained through application-specific help documents, on-line tutorials or FAQs, or through seeking help from experienced users.

Certain tasks may also be peculiar to particular work groups, with no single individual charged with documenting procedures; "best practices" may simply reside in the minds of experienced individuals or in informally authored text documents or web pages. Even where expert-authored documentation does exist, it is often not up-to-date, because it is difficult for documentation authors to keep pace with the bureaucratic, organizational, personnel, and software changes that are constant in most mid-to-large companies.

An alternative to documenting procedures is to automate them. Complex, but frequently performed tasks, such as installation of individual software packages on a single machine, are particularly well-suited to the use of scripts or wizards for automation. However, we note that there are situations where documentation is more practical than automation. For example, the task may not be preformed sufficiently often to amortize the cost of producing an automation module. Also, some tasks need monitoring by a human, to evaluate complex conditions and make high-value, non-automatable decisions, or to permit the human to learn a process which will need to be specialized in the future. Thus, documentation will continue to play an important role, even as better and more sophisticated automation tools become available.

Communities of users from whom detailed task-specific documentation is high-value include systems administrators, computer consultants, and software developers. In this paper, we will focus on the last group, as representative of the types of users who need to quickly develop documentation to communicate workgroup-specific procedures. Tasks for software developers include setting up development environments, configuring workspaces, and aligning project parameters between team members. The need for communicating such "how-to" information has become essential due to growth in the number of distributed teams who coordinate their project efforts through virtual meetings and shared online repositories.

Developers face a number of hurdles both in producing and in consuming how-to documentation, including the lack of adequate tool support. Currently, the tool most commonly used by developers for generating documentation is a standard word processor, such as Microsoft® Word [6, 15]. Using these non-specialized tools, creation of documentation is time consuming, and maintenance throughout the lifetime of a project difficult. Indeed, because of the high costs of creating documentation, it is likely that many procedures are communicated through ephemeral and non-sharable methods such as phone conversations and instant messaging sessions.

Documentation, once produced, can also be difficult to follow accurately. Users may struggle to locate GUI elements in the application interface based on textual (or graphical) descriptions in the document. In addition, users may have difficulty keeping track of their current position in the documentation. This is especially evident in digital documents where placeholders and other marking techniques are often absent. Finally, due to the lack of interactivity, users may have difficulty navigating through long and complex documents, particularly when there is a need to evaluate and traverse the correct branch in conditional statements [11].

We have developed a system, DocWizards, that enables document generation through the use of programming-by-demonstration (PBD) techniques. Through automatic capture of user interactions with an application GUI, documentation can be created simply by demonstrating the procedure. Once the procedure has been captured, semantic content can be added through operations such as inserting comments, grouping steps into semantically related sub-procedures, and parameterizing steps to generalize inputs. Addtionally, by demonstrating different action sequences to be performed when the application is in different states, the author can create structure, such as conditional statements, in the procedure.

The output from these demonstrations is a new class of documentation that we call follow-me documentation. Follow-me documentation is a form of guided walkthrough that helps users keep their place in the task both by highlighting in the documentation the next step to be performed, as well as the associated GUI elements in the actual application. Furthermore, conditional statements in the documentation are automatically evaluated and the user is directed along the correct conditional branch [1].

This paper details a study of these techniques for improving documentation practices. The study addresses two main questions: "Can PBD techniques facilitate the process of creating how-to documentation?" and "Can follow-me documentation facilitate the process of using documentation to perform tasks?"

In this paper, we describe related efforts in this field, then outline the PBD and guided walkthrough technologies utilized by DocWizards. This is followed by a description of an evaluation in which we compare use of DocWizards by developers against current documentation practices for creating and using documentation. We conclude with a discussion of the study's results and present ideas for further work in this area.

## RELATED WORK

Our work is primarily related to two areas of research: programming-by-demonstration and guided-walkthrough systems. We are not aware of work that combines both these approaches in an effort to enhance documentation.

Programming-by-demonstrations (PBD) is a well established field, and we refer the interested reader to the classic references [3, 13]. Much of the work this PBD has focused on reducing the overhead of performing repetitive tasks, typically for restricted applications, such as HyperCard [3] or text editors [12]. PBD has also been used as the main tool to create simple applications [14]. More related to our work are uses of PBD for gathering collective procedural knowledge from multiple demonstrations and distributing it as executable procedure models, as in the Sheepdog system [11]. Sheepdog, however, is not suitable for documentation generation because it relies on a complex statistical model of the task that cannot realistically be converted to a human-readable format.

The idea of creating scaffolding or training-wheels has long been used to assist new users in learning a procedure or application. Recently, approaches have been developed that use subtle guidance, so that a user's attention is directed to relevant areas within an interface, while limiting extraneous interactions with non-relevant UI elements [4, 9]. Even less intrusive forms of scaffolding appears in systems that direct the user, but ultimately leave the user free to deviate from the system direction at will [8, 17, 20].

Traditional systems generally present information in a window separate from the application [7]. As a result, users tend to miss steps while switching back and forth between documentation and application, and have difficulty in locating the application GUI components described in the documentation [10].

There are some examples of systems that use recording techniques (but not PBD) to create tutorials. For example, RWD's Info Pak Simulator creates tutorials and documentation from recordings of user interactions with an application interface. The tutorials, however, only work in simulated environments, and not within live applications [18].

Finally, many systems that achieve desirable results in providing in-context application guidance, offer little to enable end-users to participate as authors of this content [2, 9]. Usually system and application experts must create the content, which limits widespread adoption of the technology.

## DEVELOPER INTERVIEWS

In order to better understand the creation and use of internal, developer documentation, we conducted seven interviews with software developers within our organization who routinely collaborate with two or more developers on the same project. We were able to distill these interviews into the following five findings:

### *1. Modern developers operate in collaborative and geographically dispersed environments*

The developers we interviewed collaborated regularly with an average of six and no less than three other developers. Those collaborators were often geographically dispersed. We found that, on average, developers collaborated with others in three different locations, with only one developer working solely with on-site team members.

### 2. *There is a lack of standard ways of documenting and sharing knowledge*

Developers reported using multiple formats of instructional content that included standard, word processor-authored documentation and online team repositories, but also included more informal formats such as instant messaging and email. We also found that most documentation was not placed in a group accessible location, but rather transferred directly from one developer to another.

### *3. Much documentation is either unavailable or out of date*

Developers reported that even when documentation existed, it was likely not maintained through the development cycle and quickly become outdated. Most developers worked on projects that had undergone at least one major platform or development tool shift. These types of changes typically rendered the existing documentation obsolete. As a result, many interviewees resort to searching for external documentation online, because it is typically more current.

### 4. *Developers are also documenters*

All the developers interviewed or their colleagues authored documentation that was meant for group or customer consumption. The created documentation typically consisted of installation or configuration guides, but also included walkthroughs and instructional guides for new members of the development team.

### *5. There is a lack of specialized documentation tools for developers*

Conspicuously absent was the mention of any specialized documentation tools. Developers reported using Microsoft Word when creating more formal documentation, but also relied on ASCII text files.

## SYSTEM DESIGN AND FEATURES

To facilitate creation and utilization of documentation, we have created an instantiation of follow-me documentation that we call DocWizards. The DocWizards system is currently instrumented to work with the Eclipse Integrated Development Environment [5]. It relies on a novel learning algorithm called Augmentation-Based Learning (ABL) that supports documentation creation via a combination of demonstrations and user edits [16]. The main interface and features of DocWizards is shown in Figure 1.

### Documenting Creation from Application Demonstration

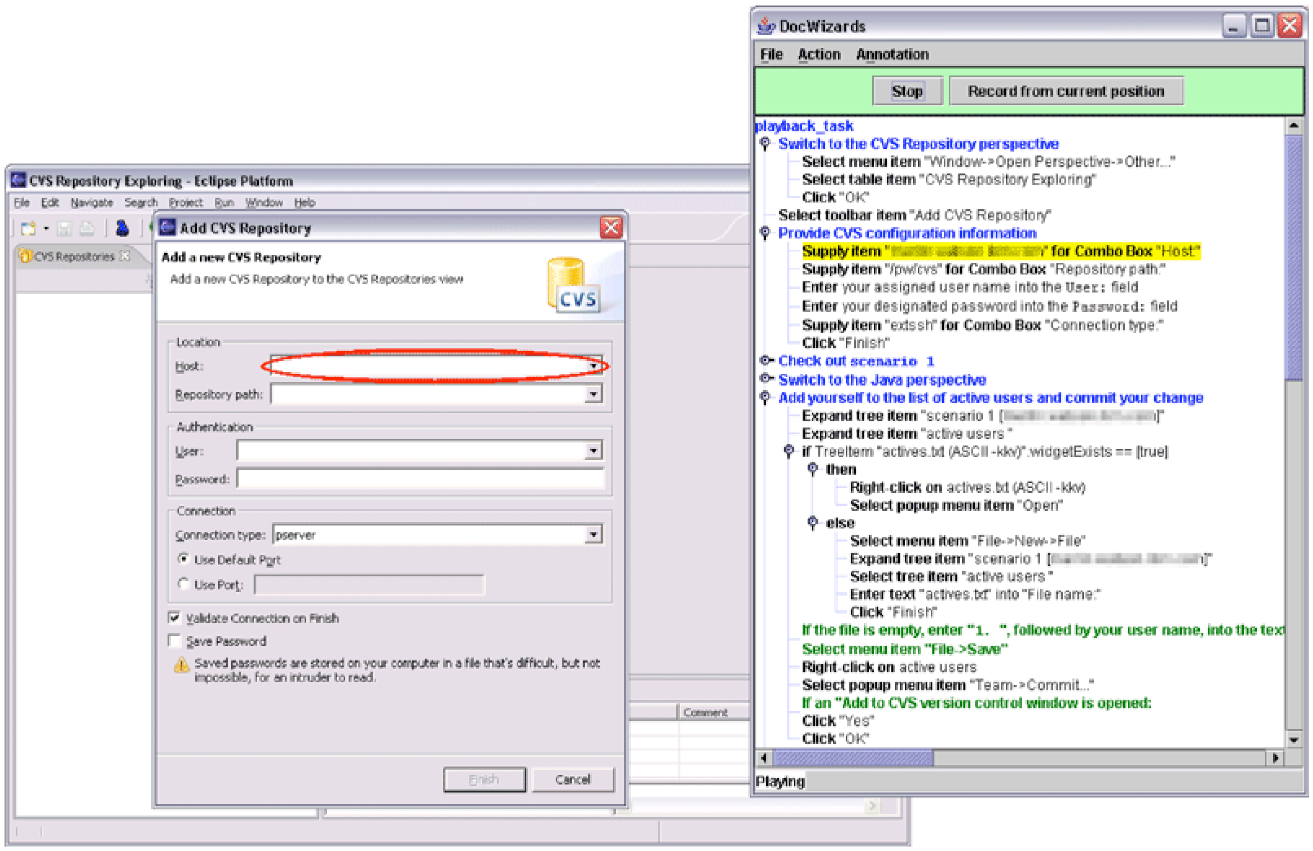
DocWizards supports the automatic, incremental creation of documentation using multiple demonstrations. When used in "recording mode" DocWizards observes a user performing actions on the application's GUI, and captures the changes in the GUI resulting from these interactions. ABL uses these observations to incrementally update a model of the task, which is displayed in human-readable form in the DocWizards user interface.

When multiple demonstrations are provided, the ABL identifies differences between the demonstrations and introduces control structures, such as if-then-else statements, to explain them. The script-like representation of the document supports limited editing operations, such as moving, copying, and deleting steps. Additional recording features available to an author include support for inserting steps at a specified location, and for accepting recording of partial procedures.

### Annotation of Captured Procedure

DocWizards automatically produces a textual description of the demonstrated task that serves as starting point for the final document. This automatically generated description contains step-by-step instructions, but no semantic information such as subtasks or expected results.

DocWizards provides means for an author to add semantic information to the document. The first feature is the ability to edit the text of each step. The author can also add comments to individual steps; these appear in the document just before the step, and add information to be displayed on the GUI. Finally, the author can hierarchically group steps into semantically meaningful units, and provide a description for each group. These groups are presented within a tree structure, and each can be dynamically hidden or exposed by clicking with the mouse. Finally, the author is able to parameterize particular steps. This allows her to convert specific recorded information (for example, a password string), into a request for a general parameter when the procedure is replayed.



```
playback_task
  Switch to the CVS Repository perspective
    Select menu item "Window->Open Perspective->Other..."
    Select table item "CVS Repository Exploring"
    Click "OK"
  Select toolbar item "Add CVS Repository"
  Provide CVS configuration information
    Supply item "[redacted]" for Combo Box "Host:"
    Supply item "/pw/cvs" for Combo Box "Repository path:"
    Enter your assigned user name into the User: field
    Enter your designated password into the Password: field
    Supply item "extssh" for Combo Box "Connection type:"
    Click "Finish"
  Check out scenario 1
  Switch to the Java perspective
  Add yourself to the list of active users and commit your change
    Expand tree item "scenario 1 [[redacted]]"
    Expand tree item "active users "
    if TreeItem "actives.txt (ASCII -kkv)".widgetExists == [true]
      then
        Right-click on actives.txt (ASCII -kkv)
        Select popup menu item "Open"
      else
        Select menu item "File->New->File"
        Expand tree item "scenario 1 [[redacted]]"
        Select tree item "active users "
        Enter text "actives.txt" into "File name:"
        Click "Finish"
    If the file is empty, enter "1.  ", followed by your user name, into the text
    Select menu item "File->Save"
    Right-click on active users
    Select popup menu item "Team->Commit..."
    If an "Add to CVS version control window is opened:
    Click "Yes"
    Click "OK"
Playing
```

Figure 1. The DocWizards system (right) running in playback mode through previously developer-authored documentation. The standard Eclipse environment (left) can be seen with a red circle annotation over the GUI widget associated with the currently selected document step, which has been indicated in the DocWizards document by a yellow highlight. Within the documentation, automatically captured action steps and script logic are displayed in black, authored-supplied annotations in green, and author-supplied hierarchical groupings in blue.

### Next Step Highlighting

DocWizards provides guidance to the user by suggesting the next step to be performed throughout the course of the task. This is indicated by highlighting both the next step in the document text as well as the widgets associated with that step on the GUI of the application on which the task is being performed. In Figure 1, the step `**Supply item** "[removed for anonymity]" **for Combo Box** "Host"' is highlighted in yellow in the document, and the corresponding widget, the entry field called "Host" is circled in red.

## METHODS

In order to evaluate how the DocWizards approach compared with standard methods of developer-authored, internal documentation, we segmented the study into three phases: *documentation authoring*, *quality assessment*, and *documentation utilization*.

The *documentation authoring* phase looked at the ability of developers to create documents using DocWizards' programming-by-demonstration approach relative to a traditional word processor. We selected a word processor as the standard tool for creating documentation because our interviews showed that it is currently the mostly common tool used by developers for creating internal documentation; previous literature has validated this observance [6].

Since the documentation produced by DocWizards is qualitatively different from traditional documentation, we generated static output from DocWizards (i.e., printed documents) to enable a relative *quality assessment* between the documents authored using this tool, and those authored using a word processor.

Finally, we conducted an evaluation of the user experience when employing the follow-me features of DocWizards as opposed to traditional documentation in the *documentation utilization* phase of the study

### Documentation Authoring Study

#### *Participants*

Eight professional software developers in our organization participated in the authoring study. All of the participants are members of project teams that collaborate on software development. None of these developers had any previous experience using DocWizards to create documentation nor had they previously documented the specific procedure we used. Seven of the eight participants rate themselves as Eclipse experienced (either ‘familiar’, ‘experienced’, or ‘expert’); the other participant rates herself as a ‘beginner’. All users except the Eclipse beginner have been using Eclipse regularly for at least 1 year. All participants are experienced code developers (mean of 10.6 years). Finally, six of the eight users have previous experience in authoring documentation as a developer.

*Preparation of Study Material*

We devised a procedure by slightly altering a real and representative Eclipse task, collected from a developer during the interviews. The task consisted of performing the following steps within Eclipse: creating a CVS Repository connection; checking-out a project, modifying and committing a CVS controlled file; and configuring the Eclipse preferences required to build the project. The total procedure contained 87 distinct steps, and included a conditional that required the user to perform different actions based on the existence or absence of a file in the environment. In addition, the procedure required that users properly parameterize certain steps that captured actual login and password strings.

We recorded a screen-grab video of the procedure being performed on the Eclipe UI with a voice-over, which provides step-by-step instruction.

*Procedure*

Each participant was involved in two individual study sessions, held at least one day apart. The DocWizards session lasted one and a half hours and the word processor session lasted one hour (extra time was required during the DocWizards session to allow for the participant to learn how to author with DocWizards). During the DocWizards authoring session, participants were first given an introduction and demonstration on how to use DocWizards’ authoring functionality. In order to test for comprehension, they were asked to perform six null-tasks which accounted for all of DocWizards’ authoring features (this was not necessary with Word since all participants were sufficiently experienced). During each session, participants were asked to view the procedure video in its entirely, after which they could ask questions on anything that was unclear about the procedure they were to document. They were then asked to document the procedure in the video with the tool they were given (Word or DocWizards depending on the session), supplying groupings and annotations when appropriate. Since we were employing a within-subjects methodology, we balanced the order of use of the two tools to account for procedure learning effects. Users were asked to think out loud during the study. We ended each authoring session when the participant claimed to have completed the documentation.

| |
|---|
| **1. Structure:** The document employs an appropriate and logical structure.<br>• Does the document have appropriately segmented subtasks?<br>• Does the document’s format make it easy for a user to quickly navigate the procedure to a specific step? |
| **2. Clarity:** The document is clearly written in a way that supports comprehension.<br>• Is the language used appropriate for the intended audience?<br>• Does the phrasing of the steps make performing each step intuitive?<br>• Is the overall purpose of each subsection or step communicated?<br>• How understandable is the document, in whole?<br>• Is the terminology used throughout the document consistent? |
| **3. Accuracy:** The document accurately and unambiguously describes the correct procedure.<br>• Use the type and frequency of errors occurring in each document to score this section. The documents will already have this information in red ink annotations. |

Table 1. Judging criteria used to evaluate the authored document quality during the Quality Assessment portion of the study.

*Evaluation Metrics*

We recorded the time to completion for each authoring session. In addition, we also administered a post-session questionnaire to gain insight into participants’ subjective impressions of each tool and a post-study questionnaire to gauge participants’ overall tool preference. We also recorded any critical incidents that took place during each session, paying particular attention to any difficulties users were having with authoring using DocWizards.

**Quality Assessment**

*Participants*

Three software developers participated as judges to assess the quality of the documentation produced during the first phase of the study. These participants all are experienced with software documentation with the Eclipse IDE, and with the procedure that was being documented in the study.

*Preparation of Study Material*

Each judge was presented with an on-paper evaluation packet containing a judging instruction sheet, eight word processor documents, eight DocWizards documents, and 16 quality evaluation forms. We printed out the word processor documents in their existing form. The DocWizards procedures were outputted to a static html representation and then printed. Any errors were marked on the printed sheets by the study proctors in order to ensure that the judges noticed them. The judges did not know the factors involved in the experiment design, including the fact that the documents were generated with two different tools.

#### *Procedure*

Each judge was presented with the evaluation packet and asked to review all the documents before assessing their quality. They were asked to evaluate the quality of each document according to the documents *clarity*, *structure*, and *accuracy*. The criteria *clarity* and *structure* were assessed according to criteria listed on the evaluation sheet (see Table 1), however, *accuracy* was to be evaluated based on the frequency and severity of errors in each document, as annotated by the study proctor. Their degree of agreement to the main statements in Table 1 (using the sub-bullets as guidance) were recorded on a 5-point Likert-scale that ranged from "Strongly Disagree" (1) to "Strongly Agree" (5); a total score of each document was created by adding the three individual scores, producing a total out of a maximum of 15 points. Later, the individual scores from each judge were combined to create an aggregate score, which had a possible total of 45 points. The judges were given three days to finish the quality assessment and allowed to work at their own pace.

### Documentation Utilization Study

#### *Participants*

Eight software developers in our organization participated in the documentation utilization portion of the study. Five of the eight participants rate themselves as Eclipse inexperienced (either 'beginner' or 'novice') and have less than three months experience using Eclipse. The other three users rate themselves as Eclipse experienced (either 'familiar' or 'experienced'), and have between 1-3 years (13-36 months) experience in using Eclipse.

#### *Preparation of Study Material*

We selected the highest-scoring documents from each study group (DocWizards and word processor), based on the evaluation in the *quality assessment* portion of the study. The DocWizards document scored slightly lower (36/45) than the word processor document (41/45) in the aggregate *structure*, *clarity*, and *accuracy* metrics. Since both documents contained a few minor errors and possible ambiguities, we manually corrected them to ensure that differences in using the documents could be attributed to their presentation and not content. The word processor document was printed out and stapled. The DocWizards procedure was pre-loaded into the DocWizards application.

#### *Procedure*

Each subject participated in two separate study sessions, one using the DocWizards document, the other with the word processor document. During the DocWizards session, users were first introduced to DocWizards and given instruction on how to use it to follow a procedure. During each session, the participants were given the documentation and asked to begin when ready and to notify us when they felt they had completed the task. We were using a within-subjects methodology, so the order of DocWizards and paper documentation sessions was balanced between the participants. In addition, to prevent participants from following the procedure from memory during the second session, we told them that the second procedure was similar in character, but different than the first; we found that this instruction was sufficient to encourage participants to perform each step from the currently presented documentation. Users were asked to think out loud while following the documentation.

#### *Evaluation Metrics*

We recoded time to completion for each documentation use session. In addition, we administered a post-session questionnaire to gain insight into participants' subjective impressions of each tool used and a post-study questionnaire to gauge participants' overall tool preference. We also recorded any critical incidents that took place during the study including errors made and corrected.

### Results

We have analyzed the data using two tests: the one-sided paired t-test (parametric), and the Wilcoxon Signed Rank test (non-parametric) [19]. These tests are appropriate for the case of two related groups when each subject belongs to both groups. The paired t-test assumes that, under the null hypothesis, the differences between the values measured for each pair are independent and identically normally distributed. The Wilcoxon test assumes that, under the null hypothesis, the differences between the values are independent and identically distributed; therefore, it is more generally applicable than the t-test, and it is only slightly less powerful when the differences are, in fact, identically normally distributed.

In the *documentation authoring* study, we found that while authoring documentation using DocWizards (29:49min average authoring time) was completed more quickly than using a word processor (34:39min average authoring time), this difference was not found to be statistically significant (t-test: p=0.229, wilcoxon: p=0.273) However, analysis of the number of errors committed showed some important differences between the tools. The DocWizards group committed a total of 20 non-critical errors (errors that did not cause the resulting procedure to fail to accomplish its goal), and 10 critical errors (errors that significantly changed the resulting procedure). The word processor group committed 19 non-critical errors and 20 critical errors. While these differences in critical errors are not statistically significant at the 0.05 significance level (t-test: p=0.064, wilcoxon: p=0.094), their borderline p-values suggest that DocWizards might enable participants to create more accurate documentation.

Users were very positive about using DocWizards as a documentation-authoring tool. Seven of the eight participants "preferred" using DocWizards to a word processor with two "strongly preferring" it (one user had "no preference"). Also, most users found DocWizards easy to learn and to use. Six participants said DocWizards was "easy" to learn, but two users found it "difficult". When asked to explain why they found it difficult, both users found the recording/annotating modal interface initially difficult to grasp. Similarly, six users found DocWizards "easy" to use, with one finding it "difficult" and one claiming it was "neither easy nor difficult".

The *quality assessment* portion of the study showed that there was also no significant difference in the quality of the resulting procedures created with the two tools. We performed our analysis by separately totaling the scored from the three judges for each document for each of the three quality metrics (*structure*, *clarity*, and *accuracy*). The DocWizards group scored 71/100 in *structure*, 68/100 in *clarity*, and 76/100 in *accuracy*. The word processor group scored 83/100 in *structure*, 82/100 in *clarity*, and 80/100 in *accuracy*. Although the word processor group scored higher in all three quality metrics, testing at the 0.05 significance level failed to reject the hypothesis of no difference among the two groups in any of the three metrics (See in Table 2), although the results for clarity are not conclusive.

The *documentation utilizing* study shows faster completion of the task when using DocWizards than when using standard paper documentation. Statistical analysis shows that the difference between the time it took to follow documentation is statistically significantly faster (t-test: p=0.0017, wilcoxon: p=0.016) when using DocWizards (8:20min average) over a word processor (12:32min average). In addition, we noticed that DocWizards users committed fewer critical errors when using documentation and recovered from any errors committed more often than users using paper documentation (DocWizards users committed 4 critical errors but corrected them all, while traditional documentation users committed 6 critical errors and only corrected half of them).

Users seemed very enthusiastic about using DocWizards as a documentation tool. Six users "preferred" using DocWizards to traditional paper documentation with half of those "strongly preferring" it (one user "preferred" paper and one had "no preference"). It is interesting to note that the only user who preferred paper documentation actually completed the task faster and with less errors when using DocWizards; he felt that while DocWizards worked well for the current procedure, it may not work for all types of procedures. We also asked participants how easy the documentation made the task and how confident they were that they had completed the task correctly. Users found that DocWizards enabled them to complete the task more easily than paper documentation (DocWizards: 7 – Very Easy, 1 – Easy / Paper documentation: 1 – Very Easy, 6 – Easy, 1 – Very Difficult). Similarly, users felt more confident that they performed the task correctly when using DocWizards over paper documentation (DocWizards: 8 – Strongly Agree / Paper documentation: 4 – Strongly Agree, 2 – Neither Agree or Disagree, 2 – Disagree).

| | t-test p-values | Wilcoxon p-values |
|---|---|---|
| Structure | 0.183 | 0.188 |
| Clarity | 0.084 | 0.063 |
| Accuracy | 0.341 | 0.449 |
| Total | 0.154 | 0.094 |

Table 2. P-values for difference in quality assessment scores for *structure*, *clarity*, *accuracy,* and the total between DocWizards and word processor-generated documentation.

## DISCUSSION

Through its use of programming-by-demonstration and guided walkthrough techniques, DocWizards facilitates the authoring and use of documentation. It also provided benefits in reducing errors while authoring and following documentation. Lastly, users had a strong preference for using DocWizards to create and follow documentation over traditional methods.

Although our data showed that using DocWizards enabled only slightly faster creation of documentation, we feel that this would change significantly with increased user experience. Despite the training and null tasks, it was clear that users were not as experienced in using DocWizards as they were with using standard word processor tools. Anecdotally, we observed expert DocWizards users, not participating in the study, completing the authoring task in less than 15 minutes (as compared with 29:49 for DocWizard users and 34:39 for word processor users). Therefore, we feel that with additional DocWizards experience, users would be able to perform an authoring task significantly faster than using a word processor.

We also found that when authoring documentation with DocWizards, users committed half the number of critical errors as they did when authoring with a word processor. This is mostly attributable to a substantial reduction in spelling and syntax errors due to automated capture of widget name and type information by DocWizards.

Although not statistically significant, we did observe a difference between the two document sets in the *structure* and particularly the *clarity* metrics. The judge's comments made it clear that *structure* often scored lower for DocWizards documents due to superficial choices that we made in how DocWizards formats html output (for example, unnecessary spaces between lines which made hierarchical structures much more difficult to understand). *Clarity* suffered somewhat in the DocWizards documents due to low human-readability of automatically captured widget names and types. This problem can be readily corrected with an enhanced naming scheme.

Much of the gain in using documentation seemed to stem from the huge improvement in navigation time when using DocWizards' guided walkthrough highlighting. Most users were observed navigating purely through the on-application highlights, which relieved them from the need to map between textual descriptions of widgets and their location on the application. Furthermore, depending on the type of widget that needed to be manipulated, the on-application highlights were sometimes sufficient to guide users without having to refer to the document; for example, circling a radio button or checkbox is sufficient to indicate both the widget and the action that needs to be performed.

Another interesting finding was the reduction of errors that users committed while using DocWizards to complete a procedure. Users were more likely to notice performing an incorrect action or skipping a crucial step, than with traditional documentation. We feel this benefit is largely a side-effect of the next-step highlighting. Because the highlighted step in the DocWizards window would not advance until the step had been performed correctly, users often noticed and recovered from accidental actions that deviated from the procedure proscribed in the document. In contrast, word processor users were not given any analogous feedback on improperly performed steps. In essence, the next-step and on-application highlights provided the application some semantic awareness, which could inform the user of syntactically valid, but erroneous inputs.

Finally, we noted an even split between two different styles of using annotation and editing features. Some users (we call them *immediate annotators*) preferred to perform actions such as adding comments, parameterizing steps, and creating hierarchical groupings, immediately after recording the relevant steps. Other users (we call them *delayed annotators*) delayed these annotations activities even when the desired annotation required only one button-press; these users seemed to prefer to stay in-mode (recording vs. annotating) for discreet chunks of time instead of switching between the two continuously. Interestingly, we noted several instances where a delayed annotator made vocal mention of needing to perform an annotation, delayed that action, and ultimately forgot to complete it upon reaching the end of the procedure.

## Lessons Learned for Programming By Demonstration Systems

Our evaluation of DocWizards has provided valuable insight into how programming-by-demonstration techniques can be better employed to reduce the costs of creating and maintaining documentation.

### *Easy conditional creation through multiple demonstrations*

We found that even novice users were able to use multiple demonstrations to create conditional statements within documentation. Not only were they able to reset the demonstration start point within the DocWizards script, but were also able to correctly make the changes to their environment required to record the additional branch path (for example, removing an existing file to create the condition "if a file exist…"). Based on this experience, multiple demonstrations may provide a viable and powerful alternative to the explicit specification techniques used in many PBD systems.

### *Supporting varied annotation styles*

Differences in annotation styles indicates that there is benefit to supporting both annotation and editing during the recording process, as well as post-recording. Special care should be taken to enable delayed annotators the ability to perform more easily forgettable annotations, such as parameterization, during initial capture while minimizing interruptions.

### *Importance of immediate feedback*

By exposing the user to the underlying structure of the recorded presentation and formatting it for human-consumption, users were able to observe and reflect upon performed steps in real time. This enabled novice users to keep track of the steps they had performed which is a crucial functionality in systems that allow multiple demonstrations and partial recordings. These two operations require that users be able to quickly locate the exact step that they are supplementing, and also the context around that step. We also observed that this immediate feedback provided new users with early confidence in the system and allowed them to concentrate on demonstrating the task, obviating the need to repeatedly check the recorded documentation throughout the authoring session.

## Lessons Learned for Guided Walkthough Systems

Through employing multiple forms of guidance DocWizards was able to improve users ability to locate UI elements, and recognize and recover from errors.

### *Multiple types of guidance*

DocWizards uses multiple forms of guidance that provide aligned, but distinct benefits to the user. Highlighting the next step to be performed within the DocWizards document view enabled users to see their current place within the documentation and also understand the context (previous steps, following step, semantic groupings) around the step they were performing. After using DocWizards, while performing a series of steps during the word processor session, one user mentioned, "this part is hardest, because I have to remember which steps I've already done". The on-application highlighting (red circles) vastly improved a user's ability to locate relevant UI objects within the application; one user commented, "I'm not sure I would have known where the 'Add CVS Repository [button] was without the guide". This type of location problem was evident when participants used the paper documentation; we often heard users say, "where do I put that" or noticed them systematically searching the interface for a button. In addition, this type of highlighting served to disambiguate similarly-named UI control items and lessened the burden of context-switching between the documentation and the application.

### *Don't fix problems, merely shows users where they exist*

We also found that by having the system track the position within the procedure and display it unobtrusively, users were better able to perform every step necessary and recover from errors. In a guided walkthrough system, this type of peripheral information is remarkably effective because it plays a dual role in reassuring users that they are performing the task correctly and prevents advancing in the document if the user has performed a step incorrectly. This subtle guidance seems to enable users to drive the interaction effectively and recover from their errors.

### *Difference between guiding and teaching*

Lastly, we noticed that the benefits of employing a guided walkthrough approach to using documentation might inhibit the acquisition of procedure knowledge. Many users commented that they were simply following the on-application highlights, for example one user commented, "I'm not reading labels, I'm just doing what the system says". Here, users were not referring to the DocWizards document unless they came to a step in which the on-application highlight was ambiguous. This approach effectively renders the author-added step groupings and commenting ineffective. Since these two abilities play a significant role in adding semantic structure and causal explanation to the individual step, the tradeoff to faster completion of the procedure may be a reduction in learning the specific procedure. It is worth noting that this finding is distinct from learning an application interface and interactions, for which we believe our approach can be beneficial.

## CONCLUSION

We have presented the DocWizards system, a documentation authoring and playback tool that employs programming-by-demonstration and guided walkthrough techniques.

In a study of software developers, we have shown that a user guide for a substantial task can be authored using DocWizards with production time comparable to that using traditional documentation tools. The documentation produced was of comparable quality but with lower error rates. With additional user training, we anticipate DocWizards authoring will be substantially faster.

While executing the documented task under the guidance of DocWizards, users completed the task substantially faster than when using traditional documentation. Additionally, the error rate using DocWizards was lower. Users clearly preferred DocWizards both for authoring and playback.

These results are strongly suggestive of the utility of both programming-by-demonstration and guided walkthough techniques in an "application how-to" documentation system. Additional studies of how these techniques can facilitate document maintenance are warranted.

## FUTURE WORK

While our work with DocWizards is encouraging, there are still questions that further studies will need to address.

From personal experience, we believe that expert users of DocWizards would be substantially more efficient at producing documentation than the novice authors that participated in the user study. Gains in efficiency should be achieved both from experience and from obtaining access to a variety of authoring features that, in the interests of simplicity, were not exposed during the study. Additional studies are needed to characterize both the learning curve and the benefits of exposing the disabled features.

An important question is whether PBD technology produces documentation that is more readily maintainable than using traditional techniques. The central question is whether partial recording techniques can be readily used to update or replace sections of a procedure that have changed *while the user is performing the procedure*. The fact that participants in this study found it easy to user partial recording in creating a conditional, leads us to believe that this may be a powerful features for document maintenance. An additional lab study that looks at the ability of users to maintain a document during simulated procedure evolution should be combined with field studies to address this question.

In the study, playback users were instructed to perform the task correctly and efficiently. These users commented that they were merely following the on-application highlighting, rather than trying to understand each step in its larger semantic context; this indicates that, to use DocWizards as a teaching tool rather than a documentation/intelligent help agent, we might need a different interaction paradigm. On the other hand, our study indicates that our subtle-guidance approach may provide the appropriate scaffolding for users to more quickly understand a novel interface. Since the user study did not have any specific metrics to measure procedure retention or application understanding additional work is needed to quantify the effects of the DocWizards approach to knowledge transfer.

## ACKNOWLEDGMENTS

[Removed to preserve anonymity]

## REFERENCES

1. Bergman, L., Castelli, V., Lau, T., and Oblinger, D. DocWizards: A System for Authoring Follow-me Documentation.
2. Carroll, J.M, and Kay, D.S. (1988). Prompting, feedback and error correction in the design of the scenario

machine. International Journal of Man-Machine Studies, 28:11-27.

3. Allen Cypher, ed., Watch What I Do: Programming by Demonstration, The MIT Press, Cambridge, Massachusetts, 1993.

4 Designing Coachmarks: http://www.developer.apple.com/techpubs/mac/AppleGuide/AppleGuide-24.html

5. Eclipse IDE® website: http://www.eclipse.org

6. Forward, A. and Lethbridge, T.C. The Relevence of Software Documentation, Tools, and Technology: A Survey. DocEng 2002, McLean, Virginia. p 26-33

7. Goodall, S. Online Help in the Real World. In Proc. SIGDOC 1991, ACM Press (1991) 1-44.

8. Jackson, K., Krajcik, J. and Soloway, E. The Design of Guided Learner-Adaptable Scaffolding in Interactive Learning Environments. In *Proc CHI 1998*, ACM Press (1998), 187-194

9. Kelleher, C. and Pausch, R. 2005. Stencils-based tutorials: design and evaluation. In Proc. of SIGCHI Conf. on Human Factors in Computing Systems (Portland, Oregon, USA, April 02 - 07, 2005). CHI '05. ACM Press, New York, NY, 541-550.

10. Knabe, K. Apple Guide: A Case Study in User-Aided Design of Online Help. In Proc CHI 1995, ACM Press (1995), 286-287

11. [Removed to preserve anonymity]

12. Tessa Lau, Steven Wolfman, Pedro Domingos, and Daniel S. Weld, Learning Repetitive Text-editing Procedures with SMARTedit, in Lieberman, ed., Your Wish is My Command: Giving Users the Power to Instruct their Software, Morgan Kaufmann, 2001.

13. Henry Lieberman, ed., Your Wish is My Command: Programming By Example, Morgan Kauffman, 2001.

14. Richard G. McDaniel , Brad A. Myers, Building applications using only demonstration, Proceedings of the 3rd international conference on Intelligent user interfaces, p.109-116, January 06-09, 1998, San Francisco, California, United States

15. Microsoft Word® website: http://office.microsoft.com

16. [Removed to preserve anonymity]

17. Quintana, C., Eng, J., Carra, A. et al. Symphony: A Case Study in Extending Learner-Centered Design through Process Space Analysis. In *Proc of CHI 1999*, ACM Press (1999), 473-480

18. RWD Technologies® website: http://www.rwd.org

19. S. Siegel and N.J. Castellan, Jr., Nonparametric statistics for the behavioral sciences, 2$^{nd}$ Edition, 1998, McGraw-Hill, New York, 399 pp.

20. Wallace, R., Soloway, E., Krajcik, J. et al. ARTEMIS: Learner-Centered Design of an Information Seeking Environment for K-12 Education. In *Proc CHI 1998*, ACM Press (1998), 195-202

## BENEFITS AND CONTRIBUTIONS

Describes and evaluates combining programming-by-demonstration and guided walkthrough techniques to create live documentation. Enables more efficient and accurate creation and consumption of documentation than traditional tools.