A Survey of Software Learnability: Metrics, Methodologies and Guidelines

Tovi Grossman, George Fitzmaurice, Ramtin Attar
Autodesk Research
210 King St. East, Toronto, Ontario, Canada, M5A 1J7
{firstname.lastname}@autodesk.com

ABSTRACT
It is well-accepted that learnability is an important aspect of usability, yet there is little agreement as to how learnability should be defined, measured, and evaluated. In this paper, we present a survey of the previous definitions, metrics, and evaluation methodologies which have been used for software learnability. Our survey of evaluation methodologies leads us to a new question-suggestion protocol, which, in a user study, was shown to expose a significantly higher number of learnability issues in comparison to a more traditional think-aloud protocol. Based on the issues identified in our study, we present a classification system of learnability issues, and demonstrate how these categories can lead to guidelines for addressing the associated challenges.

Author Keywords
Software, Learning, Learnability, Usability, Think-Aloud, Question-Suggestion, Evaluation.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces – Evaluation/Methodology.

INTRODUCTION

Over the past three decades, various concepts, methodologies, and components of usability have matured into an abundant body of arguably well-accepted usability engineering methodologies [33]. However, even when developed while following the well-established methodologies, interfaces will still possess usability problems. In a recent study, Lazar et al. found that users lose up to 40% of their time due to "frustrating experiences" with computers, with one of the most common causes of these frustrations being missing, hard to find, and unusable features of the software [25].

CHI 2009, April 4–9, 2009, Boston, Massachusetts, USA.

Part of the difficulty is that interface usage requires learning. Indeed, there is a consistent agreement that learnability is an important component of usability [1, 10, 33, 40], with some arguing that it is the most fundamental usability attribute [33]. However, while an abundance of research focused on learnability has been published, the main results do not lie solely in the HCI literature, but are scattered across numerous other research fields, such as Artificial Intelligence [19], CSCW [42], Psychology [41], and Technical Communication [17]. Thus, the designer, who is interested in understanding, and subsequently addressing, the learnability issues associated with their software, is faced with a challenging task of isolating the main research results relevant to their design needs.

Furthermore, despite the consensus that learnability is an important aspect of usability, there is little consensus among researchers as to how learnability should be defined, evaluated, and improved within a user interface. With this lack of agreement and understanding, it is no surprise that software systems still present the user with learning difficulties. Thus, our first goal and contribution is to provide a thorough survey of the existing learnability research, generalize the results into consistent frameworks and taxonomies, and provide recommendations for the evaluation of software learnability. This survey leads us to the development of a new *question-suggestion protocol* for learnability evaluation, which we first describe, and then explore, in a user study. Our study revealed that in comparison to a traditional think-aloud protocol, the question-suggestion protocol is a more efficient methodology for identifying learnability issues in software.

SURVEY OF LEARNABILITY RESEARCH

The use of the term learnability as an aspect of software usability goes back as far as 1976 [26]. Some of the earliest usability research in the 1980's assessed user's learning within word processing tools [5, 30]. Since then, HCI researchers have maintained an interest in the area of learning. In the mid 90's, the word "learnability" became a popular term to describe this aspect of usability [10, 33].

In this section we survey how learning in software has been defined, measured, and evaluated. However, we do not review the particular strategies people use to learn. We direct the interested reader to pervious literature which review such strategies [8, 36].

Learnability Definitions

Based on our literature review, the least agreed upon issue, related to learnability, seems to be its definition. In 1980, Michelsen et al. defined learnability as: "The system should be easy to learn by the class of users for whom it is intended" [32]. This definition epitomizes the difficulty associated with defining learnability – what does it mean for a system to be easy to learn? Here we review the context which researchers over the past 25+ years have considered learning within software, and then organize these definitions into a unified framework.

Initial Learning

Nielsen defines learnability as a novice user's experience of the initial part of the learning curve [33], insisting that a highly learnable system could be categorized as "allowing users to reach a reasonable level of usage proficiency within a short time". While this definition indicates the general idea of learnability, it is unclear what a "reasonable level of proficiency" would be. Furthermore, it doesn't account for the learning that occurs after such a level has been reached. While Nielsen does consider "efficiency of use" as a separate aspect of usability, defined as the performance level of an expert user, the transition from "reasonable" to "expert" performance is not considered.

Despite this potential shortcoming, defining learnability based on initial user experiences is common. While less specific, Shneiderman defines it similarly as "the time it takes members of the user community to learn how to use the commands relevant to a set of tasks" [40]. Santos and Badre provide a similar definition, as "the effort required for a typical user to be able to perform a set of tasks using an interactive system with a predefined level of proficiency" [38]. Alternatively, Holzinger defines learnability as "allowing users to rapidly begin to work with the system" [18]. Although all these definitions only consider the initial learning experiences, they still all vary.

Extended Learning

A more drastic difference is seen in definitions which consider a larger scope of learning. Dix et al. provide a definition which applies to both initial and long term learning, stating it to be the "ease at which new users can begin effective interaction and achieve maximal performance" [10]. This is similar to Rieman's definition: "Minimally useful with no formal training, and should be possible to master the software" [36]. Butler also considers initial learnability and potential for efficiency, defining learnability as both "Initial user performance based on self-instruction" and "[allowing] experienced users to select an alternate model that involved fewer screens or keystrokes" [4]. A noticeable difference in Butler's definition is that it does not consider maximal performance, or "mastering" the software, it simply requires there to be ways to increase performance with experience. Bevan and Macleod provide a similar, but more general definition, stating learnability to be the "quality of use for users over time" [3].

Learning as a Function of Experience

A prevalent trend in the above definitions is that the type of user, for which the learning occurs, is specified. For example: "novice users" [33], "members of the user community" [40], "typical user" [38], "new user" [10], "with no formal training", and "experienced users" [4]. While most of these terms usually delineate between a novice or expert user group, Davis and Wiedenbeck present another relevant type of user [9]. They define "subsequent learning" as learning by a user whom is a novice to that specific software system, but experienced with a similar system. This suggests that the user will not only have the required domain knowledge, but also a general understanding of what tools and functions will be available.

Learnability Usage in the HCI community

The above definitions give indication that there is no agreed upon definition for learnability. Even those definitions which only apply to initial learning, base their definitions on differing assumptions about the user and what the important measures are (i.e. errors, usage time, etc.).

To further strengthen our observation that the HCI community has not adopted a single agreed upon definition of learnability, we surveyed all articles published in CHI and TOCHI, for which the term learnability occurred in the body of the text. While there are many other HCI publications which we could review, we felt restricting our search to these two proceedings would be sufficient for demonstrating that the definition of learnability is not agreed upon.

The survey resulted in a collection of 88 papers (76 from CHI, 12 from TOCHI), dating from 1982 to 2008. We organized the usage of "learnability" into 8 categories. Some papers had definitions which fell into more than one category, so the total number below exceeds 88.

- Used without a definition: (45)
- Generic learnability (i.e. "easy to learn"): (7)
- Generic usability (i.e. "easy to use"): (3)
- First time performance: (17)
- First time performance after instructions: (4)
- Change in performance over time: (8)
- Ability to master system: (4)
- Ability to remember skills over time: (2)

This survey again reiterates our belief that there is no well-accepted definition of learnability.

Taxonomy of Learnability

Instead of declaring a correct definition from the prior literature which we have reviewed, we have developed a taxonomy of learnability to incorporate the most relevant ideas into a single framework. We foresee this organizational structure benefiting researchers and evaluation practitioners, allowing them to isolate specific areas of learnability which are of interest, and succinctly convey their intentions.

Learnability Scope

Our review identified two main categories:

- Initial Learnability: Initial performance with the system.
- Extended Learnability: Change in performance over time.

Note that these categories can somewhat be thought of as first and second order properties. Initial learnability applies to the performance for a single usage period, while extended learnability applies to the nature of performance change over time.

"Performance" itself could have a wide range of definitions, such as completion times, error rates, or percentage of functionality understood. We will discuss these measurable "metrics" of learnability in the next section.

User Definition

A further distinction of learnability needs to be made based on the assumptions of the user skills. For example, in initial learnability, we would generally assume the user's experience with the user interface is very limited. However, there are actually a number of relevant dimensions which we should consider:

- Level of experience with computers
- Level of experience with interface
- Quality of domain knowledge
- Experience with similar software

The first three dimensions we identify correspond to Nielsen's categorization of user experience [33]. We add a fourth category to account for designers interested in "subsequent learning" [9].

Taxonomy Examples

We illustrate the full taxonomy in Figure 1. By using its various dimensions, previous definitions can be transcribed. For example, Nielsen's definition "a novice user's experience of the initial part of the learning curve" [33], would get translated to "The ability to perform well during an initial interval." Dix et al.'s compound definition "Ease at which new users can begin effective interaction and achieve maximal performance" [10], would be translated to "The ability to perform well during an initial interval and the ability to eventually achieve optimal performance, for a user with no experience with the interface". It should be noted that for definitions which only consider initial learnability, it is redundant to define the user's experience with the interface, since this experience level must be low.

Learnability Metrics (operational definitions)

Closely related to the definitions of learnability are the metrics used to measure learnability. For example, if, based on our above taxonomy, we are interested in initial performance with the system, how do we go about measuring this initial "performance"? Just as there is a lack of consensus on the definition of learnability, there lacks a set of well-accepted metrics for learnability.

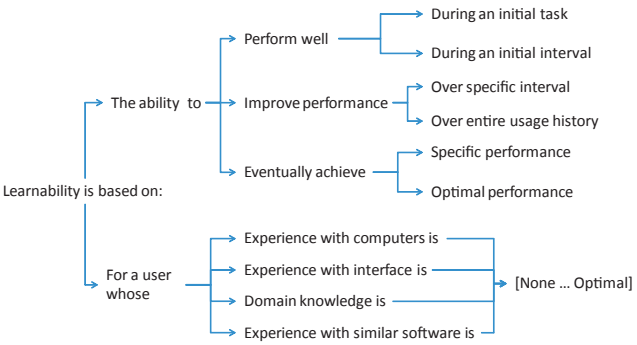

Figure 1. Taxonomy of learnability definitions

The need for such metrics is well motivated in both the software engineering [15] and human-computer interaction [4, 38] literature. From a software engineering perspective, Gilb states that "We can't possibly do adequate engineering design when we work with design objectives stated in ambiguous or abstract terms" [15]. From the HCI literature, Butler comments "We need efficient, valid, and reliable tools for measuring learnability" [4]. In an even stronger statement, Santos and Badre state "despite the consensus that learnability is an important issue in usability, few of those authors discuss at length the issue of learnability evaluation (a footnote states that they were tempted to claim that *none* have discussed the issue at length) [38].

Our survey revealed that various metrics for learnability do exist, but they are scattered across various research papers over the last two decades. We were unable to find a single collection of learnability metrics, similar to the published collections of usability metrics (e.g. [10(p239), 33(p194), 40(p135)]. We have identified seven categories of metrics which can be used for quantifying learnability (Table 1).

Since in the previous section we have defined a number of dimensions upon which learnability can be considered, it is important for us to outline how the presented metrics for learnability apply to these various aspects of learnability. In general, most of the metrics can be applied to any area of the taxonomy. One important consideration, is whether the metrics are based on a single time frame, and thus focused on initial learnability, or an extended time frame, and thus appropriate for extended learnability. However, in most cases the metrics could be easily adapted such that the relevant measure is maintained across both initial and extended learnability. For example metric T4 could be used for initial learnability by instead considering it as "number of task errors during an initial task".

Learnability Evaluation Methodologies

To use any of the metrics in Table 1, an evaluation needs to be performed to capture their values. In the usability engineering research literature, there are numerous methodologies for assessing software usability. However it is not clear which of these methodologies are most appropriate for specifically focusing on software learnability.

Task Metrics: Metrics based on task performance
T1. Percentage of users who complete a task optimally. [28]
T2. Percentage of users who complete a task without any help. [28]
T3. Ability to complete task optimally after certain time frame. [4]
T4. Decrease in task errors made over certain time interval. [32]
T5. Time until user completes a certain task successfully. [33]
T6. Time until user completes a set of tasks within a time frame. [33]
T7. Quality of work performed during a task, as scored by judges. [9]
Command Metrics: Metrics based on command usage
C1. Success rate of commands after being trained. [5]
C2. Increase in commands used over certain time interval. [32]
C3. Increase in complexity of commands over time interval. [32]
C4. Percent of commands known to user. [2]
C5. Percent of commands used by user. [2]
Mental Metrics: Metrics based on cognitive processes
M1. Decrease in average think times over certain time interval. [32]
M2. Alpha vs. beta waves in EEG patterns during usage. [41]
M3. Change in chunk size over time. [38]
M4. Mental Model questionnaire pretest and post test results. [35]
Subjective Metrics: Metrics based on user feedback
S1. Number of learnability related user comments. [32]
S2. Learnability questionnaire responses. [12, 27]
S3. Twenty six Likert statements. [12]
Documentation Metrics: Metrics based on documentation usage
D1. Decrease in help commands used over certain time interval. [32]
D2. Time taken to review documentation until starting a task. [32]
D3. Time to complete a task after reviewing documentation. [32]
Usability Metrics: Metrics based on change in usability
U1. Comparing "quality of use" over time. [3]
U2. Comparing "usability" for novice and expert users. [3]
Rule Metrics: Metrics based on specific rules
R1. Number of rules required to describe the system. [19, 24]

Table 1. Categories of learnability metrics.

A main distinction for the type of usability evaluations which can be performed is whether they are to be formative or summative [33]. A formative evaluation is used to learn about the usability problems associated with the system, in an effort to improve the interface. A summative evaluation is used to assess the overall usability of a system, in an effort to either compare to another competing system, or to determine if it meets requirements which have been set out. This distinction can also be made for learnability evaluations. Formative evaluations should expose learnability issues, while summative evaluations should provide an overall assessment of the system's learnability. In the following two sections we survey the methodologies used to perform these two types of learnability evaluations.

Formative Learnability Evaluation Methodologies

One of the most common forms of usability testing, the think-aloud protocol [13], was originally used in the HCI field to evaluate initial learnability. Mack et al. asked users to verbalize as they worked, describing questions, plans, strategies, inferences, and knowledge they were aware of [30]. The protocol was performed both during a training phase, where users studied the system documentation, and during the performance of a test task. All comments were recorded and the screen was captured by video, for post-experiment qualitative analysis. This methodology can be useful for identifying learnability issues, but takes place in the unnatural environment of a usability lab.

Rieman used a naturalistic learnability methodology, where a diary was given to participants, who kept a record of all learning related activities that occurred within a one week time period [36]. Participants were then questioned about their entries in a structured interview, and also asked about learning activities outside of the one week time frame. This methodology has the benefit of being performed in a real usage setting, but it relies on the users having the ability to identify the learnability issues which they encounter.

Another methodology of interest is the "coaching" or "question-asking protocol". This protocol was introduced by Kato in 1986 [22], but, despite being referenced in Nielsen's Usability Engineering [33], has received little attention from the HCI community. With this methodology, an expert, or "coach" sits beside the participant, and answers any questions that the participant may have during use. This protocol was originally proposed as an alternative to the think-aloud protocol, so that participants would be encouraged to verbalize their thoughts.

Mack and Robinson [29] summarize the work of Kato, and state that "question asking … creates opportunities for users to learn something about the system they are using". So, although the question-asking protocol was used for a general usability study, it seems that this would be a perfect methodology to use to identify learnability issues. Similar protocols were proposed by Franzke, where hints which were given to users were tracked and measured [14], and by Dumas and Redish, who proposed counting calls to a simulated help desk [11].

Summative Learnability Evaluation Methodologies

In a follow-up to their initial think-aloud study, Carroll et al. [5] augmented the think-aloud methodology with a quantitative analysis of the user's performance during the test task, and time spent learning the documentation.

Davis and Wiedenbeck propose a different methodology for assessing learnability [9]. As with the above methodology users are given a fixed amount of time for training, and then during a test task, users are left by themselves, with only the system documentation to refer to as aid. The users are given a fixed amount of time to complete the task, and the evaluation is then based on the final product of their task, as scored by judges. The judges scored a transcribed word processor document. This methodology may provide a more natural usage environment than the think-aloud protocol.

A more controlled method to obtain subjective learnability feedback is demonstrated by Elliott et al. [12]. They brought participants into a lab and had them complete a set of given tasks, and gave the participants a survey of 25 learnability related questions. Butler [4] uses a similar methodology, but also records the time to complete the task. Of course, based on the desired information, most of the metrics discussed in the previous section could be recorded during this type of lab-based study.

The above types of summative evaluations can be made by using any subset of the metrics outlined in the previous section, although the evaluator could choose specific metrics that are most relevant to the product service or success. For example, for software that depends on consumers' initial impressions, *task metrics*, capturing initial learnability could be used, such as *T2* or *T5*.

Initial and Extended learnability evaluation methodologies

In relation to our taxonomy, the majority of the evaluation methodologies which we reviewed were focused on *initial learnability*. This is likely a reflection of the fact that in general, evaluating initial learnability will be an easier task. Nielsen states: "One simply picks some users who have not used the system before and measures the time it takes them to reach a specified level of proficiency in using it" [33].

Evaluating extended learnability is a more challenging task, as it can require the assessment of performance over a long period of time. To do this completely accurately the study would need to follow a group of users of the desired period of time, and potentially take years to complete.

Bevan and Macleod suggest comparing usability for novice and experienced users [3]. With a large enough sample size, and an adequate range of user experience levels studied, this type of study could reveal important information on how user performance progresses over long periods of time. Recent work has also looked at assessing user "expertise" [20, 37]. Such a metric could be used when comparing users at various experience levels, to get a sense of the progression of performance over time. However, a potential problem with this methodology is that the result may be misleading, in that they would reveal a continuous progression of performance improvement over time, where for each individual user this progression could be extremely discrete, with major learning events occurring periodically.

THE QUESTION-SUGGESTION PROTOCOL

Many of the methodologies discussed in our survey take on the same form as traditional usability studies. If the goal of the study is to identify learnability issues which can be addressed, it is not clear if using traditional usability methodologies is the right approach, since they may not be designed to specifically expose learnability problems.

We identified the coaching protocol as a particularly interesting methodology which may be suitable for highlighting learnability issues. That is, the dialogue which occurs between the user and coach will identify areas of the system which present learnability challenges.

However, as with the other discussed methodologies, the coaching methodology is focused on initial learnability, as the user is only coached when they are stuck with the interface. To truly understand extended learnability, we must also understand what causes users to not just acquire new abilities, but what causes them to improve their usage behaviors by finding more efficient strategies.

As such we suggest augmenting the question-asking protocol [22] into a "question-suggestion" protocol, in which the expert can also freely provide advice to the user. This would replicate a scenario, where a user is performing the task next to a colleague, and the colleague notices a usage behavior which could be improved upon. This type of informal, or "over the shoulder" learning has been shown to be a common way for users to learn [31, 42]. Including suggestions into the protocol would allow the system evaluators to identify causes for suboptimal performance, indicating barriers to *extended learnability*. Furthermore, allowing a coach to provide suggestions may allow the user to progress further through a task, which in turn could expose a larger set of learnability issues. Thus, we believe the question-suggestion protocol could be a suitable methodology for learnability evaluation, since it captures both the initial and extended learnability dimensions of our taxonomy. By selecting appropriate users, the methodology can also capture the user dimension of the taxonomy (Figure 1).

USER STUDY

While previous work has explored the coaching protocol, our literature review has not identified a previous study testing the question-suggestion protocol which we have described above. Furthermore, we are unaware of previous work which has compared any such "coaching" protocol to the traditional think-aloud protocol, to explore the relative benefits of each, and the nature of the observations which each will produce. In this study, we explore this potential learnability evaluation methodology.

To perform the study we needed a software application which would be complex enough to possess numerous learnability challenges, and also a system for which we would have access to a system expert that could serve as a coach. For the purpose of the evaluation, we chose the popular computer-aided design system, AutoCAD.

Participants

Ten volunteers (7 male, 3 female) participated in the experiment. Participants were university architecture undergraduate students, aged 19-30. Participants were chosen with usage experiences ranging from two months to five years. We decided it would be inappropriate to seek users with no AutoCAD experience, since the software is not meant to be walk-up and use. Referring to our taxonomy, it would be inappropriate to base the learnability of AutoCAD on an "initial interval" or "initial task". Further, architecture students would have the quality of domain knowledge which would be required to successfully complete tasks with AutoCAD. We did not control for the participants' experience level with similar software

The Coach

We recruited a single AutoCAD expert to act as the coach. The coach was a professional architect with 12 years of experience using AutoCAD, and had taught AutoCAD at the university level for 4 years.

Setup

The study took place in a usability lab. The participant sat at a standard PC workstation. The screen video was captured directly and audio was captured through an external microphone. Each subject experienced a think-aloud protocol and a question-suggestion protocol while being given a series of real-world AutoCAD tasks to perform. In the think-aloud sessions, the "coach" sat further away from the participant, at a table where they could still observe the user's actions, and in the question-suggestion sessions the coach sat beside the participant. We found in a pilot study that this helped the user differentiate between the two protocols, so they would not ask for help in the think-aloud protocol sessions.

In addition to the coach, an experimenter sat in the room to explain the study and procedure to the participant. The experimenter was an HCI researcher. The presence of the experimenter was necessary to ensure the study protocols and procedures were being followed correctly, as the coach did not have experience with performing usability studies.

Task Scenarios and Procedure

During the design of the study, the coach generated 4 different real-world task scenarios for use in the studies, and template data for use during these sessions. The task scenarios were purposely made to be high-level, requiring numerous steps to be carried out, in potentially a number of different ways, covering a wide range of the application's functionality. The tasks were generated to be at a moderate level of difficulty, so that it would not be too difficult for the most novice users, and not too easy for the most experienced participants. Our hope was that these scenarios would expose both cases where users did not know immediately how to complete the task (exposing initial learnability issues), and where users knew of a way to complete the task, but not necessarily an accurate or efficient way (exposing extended learnability issues).

The experiment occurred in two 30 minute sessions, one under the think-aloud protocol, and one under the question-suggestion protocol. Table 2 shows the instructions given to the participant and coach for the two sessions. In the question-suggestion protocol, the coach was instructed to only provide suggestions when he felt it would provide an immediate benefit and be welcomed by the user.

Along with the instructions provided in Table 2, users were asked to complete the tasks as accurately as they would in a real-world scenario, as if it were an ongoing project that they would be using in the future. Users were also told they could ask task clarification questions at any time, regardless of the protocol currently in use.

Once the instructions were given, users were given one task scenario at a time to complete, each lasting at most 15 minutes. Two tasks were performed in each session. The user was asked to proceed to the next task if not finished after 15 minutes, or to stop working if they had not completed a task at the end of a 30-minute session.

Think-Aloud Protocol Instructions to Participant:
1. Try to verbalize all of your actions, intentions and thoughts during the study. I will give you an example.
2. Focus on getting the task done, as you would in the real world.
3. You may be asked to do things you've never done before, or don't know how to do. Try to do your best to figure out how to accomplish these tasks, as if you were at home by yourself. You may use help systems, online searches, or any other resources that you would normally use.
4. The coach is only here to make observations, he will clarify any aspects of the tasks, but you cannot ask him how to accomplish the task.
Think-Aloud Protocol Instructions to Coach:
1. If the user is frustratingly stuck, provide specific procedural suggestions, to get the experiment moving again.
2. Do not tutor, or explain at length.
Question-suggestion Protocol Instructions to Participant:
1. Ask relatively specific, procedural questions.
2. Try to answer your own questions first, but do not engage in extensive problem solving.
3. Focus on getting the task done, as you would in the real world.
Question-suggestion Protocol Instructions to coach
1. Reply with specific procedural answers, to the underlying form of the question.
2. Do not tutor, or explain at length.
3. Maintain focus on the task, even when providing suggestions.
4. Provide suggestions if you notice inefficient usage behaviors, and if the suggestions would likely be welcomed and beneficial to the user.

Table 2. Protocol instructions for the participant and coach.

Design

A mixed factorial design was used for the study, with the within subject variable being the evaluation protocol (think-aloud, question-suggestion), and the between subject variable being experience (ranging from 2 months to 5 years experience). Although we didn't preemptively control for experience level, it turned out that we could group user experience at four different levels: 2 months (2 users), 1.5 years (2 users), 2 years (2 users) and 5 years (4 users). The order of the protocols was counterbalanced, with half of the participants performing each protocol first. The tasks were completed in the same order for each participant.

Results

Observed Learnability Issues

Similar to previous work comparing evaluation methodologies [21], our main dependant variable of interest was the number of problems discovered, specifically learnability issues. Although there are issues with relying on "problems detected" for comparing methodologies [43], we felt it to be most appropriate for our purposes.

Learnability issues were recorded by the experimenter, and *not* the coach. We felt that allowing the coach to record learnability issues would possibly distract him from properly performing the evaluation protocols. Furthermore, allowing the coach to record learnability issues would be slightly misleading, since most often such skilled software experts are not present during think-aloud usability studies.

In most cases, we could rely on the dialogue to identify learnability issues. However, in some cases, users were

clearly having difficulties learning to use an aspect of the system without verbalizing those difficulties (even in the think-aloud protocol). Such events were also recorded as learnability issues. For the dialogue, the experimenter judged whether or not the user's or coach's utterances were a result of a learnability issue. If this was deemed to be the case, the learnability issue would be recorded.

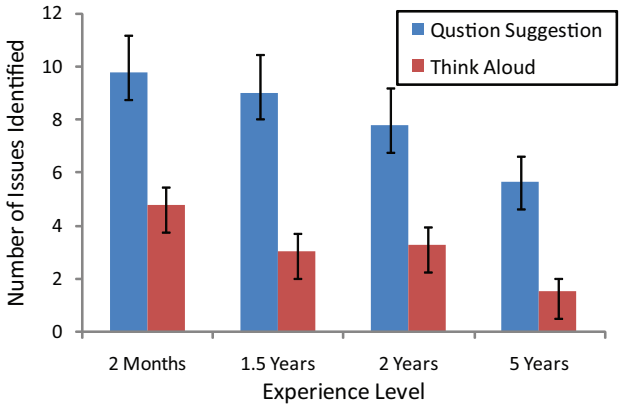

Figure 2. Average number of learnability issues identified by protocol, measured over each 15 minute task.

A repeated measure analysis of variance, on the number of learnability issues which occurred per 15 minute task, showed significant main effect for Protocol ($F_{1,6} = 48.8$, $p < 0.001$) and Experience ($F_{3,6} = 5.12$, $p < 0.05$). The average number of learnability issues reported were 2.8 for the think-aloud protocol and 7.55 for the question-suggestion protocol. This is an important result, as it shows that the question-suggestion protocol accomplished exactly what we had hoped, as more learnability issues were identified.

For both protocols, the number of learnability issues reported decreased as the user experience level increased (Figure 2). The interaction between Protocol and Experience was not significant. This was surprising, as one could expect that the difference between the two protocols may depend on the experience level of the users.

Categorization of Learnability Issues

From the collection of learnability issues which were identified, we were able to identify and categorize specific types of problems which occurred (Table 3). The categorization that we decided upon is as follows:

- Understanding Task Flow. Often it was clear to the user that a series of operations would be required to complete a certain aspect of their task, but they did not know where to start, or the high-level task flow to follow.
- Awareness of Functionality. Another typical problem was that users were not aware of a specific tool or operation which was available for use.
- Locating Functionality. This problem occurred when the user was aware of a certain operation which the system possessed, but could not figure out where to find it in the user interface, so that it could be utilized.
- Understanding Functionality. This problem means that users were aware of a single, specific, tool or function, able to locate it, but could not figure out how to use it.
- Transitioning to Efficient Behaviors. This issue is more specific to extended learnability. It indicates the user was aware of certain functionality, but chose not to use it.

Category	Example Learnability Issue Observed
Task Flow	Did not know how to set up a text style. Did not know how to evenly space 6 objects.
Awareness	Was not aware of the *divide* command. Was not aware of the *mirror* command.
Locating	Could not find *layer manager* icon. Could not find *leader* toolbar.
Understanding	Couldn't figure out how to use the *extend* command. Couldn't figure out how to use the *align* command.
Transition	Did not use *match properties* tool and instead created a new hatch. Didn't think to use *mirror* command, and instead manually created a section of a plan.

Table 3. Categories of observed learnability issues.

In Figure 3, we outline the relative proportions of these categories that each of the protocols exposed. It can be observed that the think-aloud protocol identified a higher proportion of locating issues, while exposing a smaller proportion of awareness issues and no transition issues. This is a result of how think-aloud protocols are performed. It is more difficult to identify awareness and transition issues, since it is unlikely the user would verbalize these issues, and they may not expose any identifiable behavior indicating that they are having difficulties. These types of problems are identifiable in the question-suggestion protocol, highlighted when the coach gives the user an associated suggestion (e.g. "Did you know that you could use a divide command", or "why don't you use the match properties tool instead, it will save you time and effort").

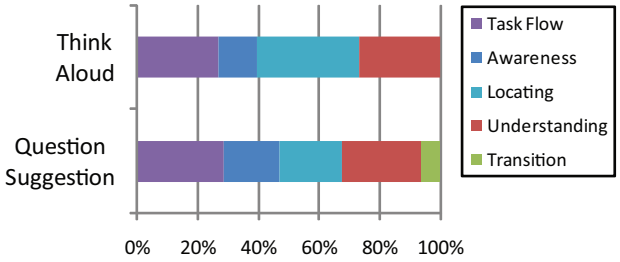

Figure 3. Proportion of issues observed by category.

Figure 4 shows the average number of problems identified in each category, as a function of the user's experience level. The most prominent and divergent effect here is that Awareness was a major issue for the most novice users, and that Task Flow was rarely a problem for the expert users. This indicates that the experienced users had a sound understanding of the general method used to accomplish high level tasks, even if they may not have known exactly how to accomplish it, step by step. It can also be observed that no transition issues were experienced by the most novice level of users. This is consistent with the question-suggestion protocol instructions, which were given to the coach, since suggestions about transitioning to expert

behaviours would likely have been more of a hinderence at the novice's early usage stage. Refering back to our taxonomy, Figure 4 also indicates that each learnability category can expose issues related to both initial and extended learnability. The one exception is the Transition category, which would mostly expose issues related to extended learnability.

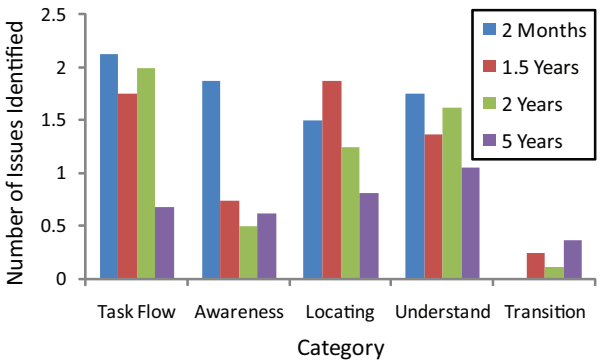

Figure 4. Identified categories by experience level

Questionnaire Results

After the experiment we administered a questionnaire, asking the participants about their opinions of the two evaluation protocols. The questionnaire consisted of 6 questions on a 5-point Likert Scale, for each of the protocols. The results were analyzed using a paired t-test. The protocol had a significant effect on three responses.

First, the response to the user's ability to progress quickly through the task was more positive for the question-suggestion protocol ($t_9 = 2.60$, $p < .05$). This is an important result, as it shows that users were able to accomplish more in the question-suggestion protocol. This is one reason the protocol was able to identify more learnability issues.

Second, the response to getting frustrated during the evaluation was lower for the question-suggestion protocol ($t_9 = 2.63$, $p < .05$). Although this was not the intention of the new protocol, it is a beneficial side-effect.

And third, the response to learning new things was more positive for the question-suggestion protocol ($t_9 = 2.45$, $p < .05$). This indicates that the coach was successfully giving useful suggestions to the users, and not simply stating redundant information.

GUIDELINES FOR INTERFACE LEARNABILITY

A potential implication of identifying the above-mentioned learnability categories is that design guidelines for improving learnability which isolate these specific categories can be developed. Although our categorization is based on a study with a single specific software application, we believe the identified categories can be generalized to other software applications. In this section, we discuss example guidelines for addressing each of the identified categories. These are not meant to be exhaustive; they only exemplify how our categorization has the potential to identify future research directions.

Task Flow

Improving a user's understanding of task flow may be one of the most difficult challenges to address. Even with a shift towards task-based documentation [6], the queries a user would enter are still generally low level tasks. For example, in an image editor, one may search for "create a new brush style", but it is unlikely they would find help if they searched for "create a texture that looks like a hardwood floor". Allowing users to locate such higher level task flows is a potential guideline which could be investigated.

Awareness

The graphical user interface is an excellent way of making functionality visible to the user, and thus improving their awareness through visual exposure [39]. However, in a command rich interface, it is impossible for a user to be aware of everything. There are some interesting areas to explore for addressing this design guideline. Previous efforts to implement intelligent agents, such as Microsoft's "Clippy", which present relevant information to the user, have failed [44]. However, we do not believe the problem is unsolvable. There is still room to explore strategies for making such suggestions *less intrusive*, and *more semantic*, so that users are gradually made aware of information which is likely to be useful. For example, Grossman et al. showed that by presenting an audio hint for a hotkey when a user selected an associated menu item, users were more likely to transition to this more expert behavior [16].

Locating

The typical approach to aiding the user's ability to locate functionality is to provide online documentation. Recently, stencil-based tutorials, which illustrate directly on the user interface, were shown to be an effective way of identifying the location of functionality [23]. This form of contextual aid is extremely important for applications with customizable UI views. In our study, we observed, on numerous occasions, users disabling and immediately re-enabling toolbars so they could locate them on the screen layout. Thus, a potential design guideline which addresses locating is to make system documentation sensitive to the user's current UI layout.

Understanding

As with locating functionality, help systems are often the method of choice for improving the understanding of how to use functionality. However, complex operations can be difficult to describe to the user, without providing an actual demonstration. As evidence, the coach in our study mentioned that on numerous occasions he wished he could drive the application for the user, since it would convey the ideas much more efficiently than trying to describe them. Two relevant guidelines can be taken from this. First, is to explore help systems which guide by demonstration, in the context of the user's own interface. Second, is to enrich documentation with multimedia content [34], so that complex operations can be clearly demonstrated to the user.

Transitions

It is fairly well understood that user performance plateaus at a suboptimal level of performance [8, 33]. Researchers have proposed training wheel interfaces, so that advanced functionality is gradually exposed to users [7], however, in many cases users are fully aware of the existence of functionality which can help them, and do not use it [8]. Recently, it was proposed that software systems could increase the user's motivation to transition to efficient behaviors [16]. This is a design guideline that provides an interesting new space to explore. It is likely that users perform a mental "cost benefit" analysis before deciding to try new functionality. We need to better understand and attempt to model this calculation that users perform, and then seek design strategies for biasing the decision towards what would be the optimal behavior.

DISCUSSION AND FUTURE WORK

In the first part of this paper we presented a survey of software learnability. It is important to clarify that our literature review, including the development of the taxonomy, was based on a review of research relating to the underlying concept of learning in software, and not strictly on the term "learnability". We did find that the majority of papers which label the higher level concept with a single term do use the word "learnability" which is why we have used this term throughout the paper. While our taxonomy is mostly aimed at researchers, future work could translate the taxonomy into a formal specification based language, which could increase its practical importance.

In the second part of the paper we investigated a new question-suggestion evaluation protocol, which was shown to have benefits in comparison to the more traditional think-aloud protocol. The results should be treated with caution, since this was an initial study with only ten participants, and the learning issues were identified by the experimenter without external validation. Our intention is not to find a replacement to think-aloud, or even to proclaim that the question-suggestion protocol is a "better" protocol to be used for usability evaluations. The think-aloud protocol has some important benefits which make it appropriate for certain types of studies. For one, the think-aloud protocol does not require the presence of a software expert, as the question-suggestion protocol does, which could add complications to performing a study. Furthermore, think-aloud studies allow observers to see how well users can recover from errors, and how long it takes them to figure things out on their own. These issues are important to understand before software is released, and the question-suggestion protocol does not provide such information. Finally, the suggestions provided in the question-suggestion protocol accelerate the natural learning process, and so the study is not representative of natural software usage.

One of the main benefits of the question-suggestion protocol was the presence of a software expert, who acted as the coach. The expert was able to identify inadequate behavior, which an evaluator less educated in the system may not recognize. Thus, having an expert present allows the evaluator to identify extended learnability issues, which would otherwise be ignored. This exposes an advantage in comparison to not just the think-aloud protocol which we compared to, but the other formative methodologies which we reviewed. Thus, we feel the study would provide similar results if we had compared to other methodologies. This indicates that other protocols could be explored which rely on the presence of an expert.

We also feel that there is room for the development of more elegant metrics which measure learnability. The metrics which have been developed to date are generally measured during specific tasks and thus cannot be generalized or compared to other tasks or user interfaces. For example, there are no generalized metrics which would allow a designer to determine if their interface, overall, increases learnability, in comparison to another. Efforts in developing such "universal" usability metrics have been conducted [27], and similar efforts could be made towards learnability.

CONCLUSIONS

In conclusion, we have performed an investigation into the issues relating to the learnability of software applications. A significant contribution of our work is the collection of these results into a survey of definitions, metrics, and methodologies used to evaluate learnability. We also developed a new question-suggestion evaluation protocol to specifically identify existing learnability issues in an application, and compared it to a traditional think-aloud evaluation. Not only did this study show that our new methodology exposed a significantly larger number of learnability issues, it also allowed us to collect and categorize learnability issues. We believe that there is still room for refinements to the evaluation process of software learnability and a vast design space which remains to be explored for improving learnability. Our hope is that the current work will serve as a resource and inspiration for future research in these areas.

ACKNOWLEDGEMENTS

We would like to thank the CHI reviewers who provided particularly thorough reviews and thoughtful suggestions, which were incorporated into the final version of this paper.

REFERENCES

1. Abran, A., Khelifi, A., Suryn, W. and Seffah, A. (2003). Usability Meanings and Interpretations in ISO standards. *Software Quality Journal*. 11:325-338.
2. Baecker, R., Booth, K., Jovicic, S., McGrenere, J. and Moore, G. (2000). Reducing the gap between what users know and what they need to know. *Proceedings on the 2000 conference on Universal Usability*. 17-23.
3. Bevan, N. and Macleod, M. (1994). Usability measurement in context. *Behaviour and Information Technology*. 13:132-145.
4. Butler, K. A. (1985). Connecting theory and practice: a case study of achieving usability goals. *ACM CHI*. 85-88.

5. Carroll, J., Mack, R., Lewis, C., Grischkowsky, N. and Robertson, S. (1985). Exploring exploring a word processor. *Human-Computer Inter*. 1:283-307.
6. Carroll, J. M. (1990). The Nurnberg Funnel. MIT Press.
7. Carroll, J. M. and Carrithers, C. (1984). Training wheels in a user interface. *Comm*. *ACM*. 27(8):800-806.
8. Carroll, J. M. and Rosson, M. B. (1987). Paradox of the active user. *Interfacing thought: cognitive aspects of human-computer interaction*. MIT Press. 80-111.
9. Davis, S. and Wiedenbeck, S. (1998). The effect of interaction style and training method on end user learning of software packages. *Interacting with Computers*. 11(2):147-172.
10. Dix, A., Finlay, J. E., Abowd, G. D. and Beale, R. (2003). Human Computer Interaction. Prentice-Hall.
11. Dumas, J. F. and Redish, J. C. (1993). A Practical Guide to Usability Testing. Greenwood Publishing Group Inc.
12. Elliott, G. J., Jones, E. and Barker, P. (2002). A grounded theory approach to modeling learnability of hypermedia authoring tools. *Interacting with Computers*. 14(5):547-574.
13. Ericsson, K. A. and Simon, H. A. (1984). Protocol Analysis: Verbal Reports as Data. MIT Press.
14. Franzke, M. (1995). Turning research into practice: characteristics of display-based interaction. *ACM CHI*. 421–428.
15. Gilb, T. (1996). Level 6: Why we can't get there from here. *IEEE Software*. 13(1):97-98, 103.
16. Grossman, T., Dragicevic, P. and Balakrishnan, R. (2007). Strategies for accelerating on-line learning of hotkeys. *ACM CHI*. 1591-1600.
17. Haramundanis, K. (2001). Learnability in information design. *ACM SIGDOC*. 7-11.
18. Holzinger, A. (2005). Usability engineering methods for software developers. *Comm. of ACM*. 48(1):71-74.
19. Howes, A. and Young, R. M. (1991). Predicting the learnability of task-action mappings. *ACM CHI*. 1204-1209.
20. Hurst, A., Hudson, S. E. and Mankoff, J. (2007). Dynamic detection of novice vs. skilled use without a task model. *ACM CHI*. 271-280.
21. Jeffries, R., Miller, J. R., Wharton, C. and Uyeda, K. (1991). User interface evaluation in the real world: a comparison of four techniques. *ACM CHI*. 119-124.
22. Kato, T. (1986). What question-asking protocols can say about the user interface. *Int. J. Man-Mach. Stud.* 25(6):659-673.
23. Kelleher, C. and Pausch, R. (2005). Stencils-based tutorials: design and evaluation. *ACM CHI*. 541-550.
24. Kieras, D. E. and Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*. 22:365-394.
25. Lazar, J., Jones, A. and Shneiderman, B. (2006). Workplace user frustration with computers: An exploratory investigation of the causes and severity. *Behaviour and Info. Technology*. 25(3):239-251.
26. Licklider, J. C. R. (1976). User-oriented interactive computer graphics. *ACM UODIGS*. 89-96.
27. Lin, H. X., Choong, Y. and Salvendy, G. (1997). A proposed index of usability: a method for comparing the relative usability of different software systems. *Behaviour & Information Technology*. 16:267-278.
28. Linja-aho, M. (2005). Evaluating and Improving the Learnability of a Building Modeling System. Helsinki University of Technology.
29. Mack, R. and Robinson, J. B. (1992). When novices elicit knowledge: question asking in designing, evaluating, and learning to use software. *The psychology of expertise: cognitive research and empirical AI*. Springer-Verlag, Inc. p. 245-268.
30. Mack, R. L., Lewis, C. H. and Carroll, J. M. (1983). Learning to use word processors: problems and prospects. *ACM Trans. Inf. Syst.* 1(3):254-271.
31. Mackay, W. E. (1991). Triggers and barriers to customizing software. *ACM CHI*. 153-160.
32. Michelsen, C. D., Dominick, W. D. and Urban, J. E. (1980). A methodology for the objective evaluation of the user/system interfaces of the MADAM system using software engineering principles. *ACM Southeast Regional Conference*. 103-109.
33. Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.
34. Palmiter, S. and Elkerton, J. (1991). An evaluation of animated demonstrations of learning computer-based tasks. *ACM CHI*. 257-263.
35. Paymans, T. F., Lindenberg, J. and Neerincx, M. (2004). Usability trade-offs for adaptive user interfaces: ease of use and learnability. *ACM IUI*. 301-303.
36. Rieman, J. (1996). A field study of exploratory learning strategies. *ACM TOCHI*. 3(3):189-218.
37. Santos, P. J. and Badre, A. N. (1994). Automatic chunk detection in human-computer interaction. *AVI*. 69-77.
38. Santos, P. J. and Badre, A. N. (1995). Discount learnability evaluation. *GVU Technical Report GIT-GVU-95-30*. Georgia Institute of Technology.
39. Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. *Comp*. 16(8):57-69.
40. Shneiderman, B. (1997). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman Publishing Co., Inc.
41. Stickel, C., Fink, J. and Holzinger, A. (2007). Enhancing Universal Access–EEG Based Learnability Assessment. *Lecture Notes in Comp. Sci*. 813-822.
42. Twidale, M. B. (2005). Over the Shoulder Learning: Supporting Brief Informal Learning. *CSCW*. 14(6):505-547.
43. Wixon, D. (2003). Evaluating usability methods: why the current literature fails the practitioner. *Interactions*. 10(4):28-34.
44. Xiao, J., Stasko, J. and Catrambone, R. (2004). An empirical study of the effect of agent competence on user performance and perception. *Autonomous Agents and Multiagent Systems - Volume 1*. 178-185.