HUMAN–COMPUTER INTERACTION, 2015, Volume 30, pp. 64–102

ISSN: 0737-0024 print / 1532-7051 online
DOI: 10.1080/07370024.2014.881668

Taylor & Francis
Taylor & Francis Group

An Investigation of Metrics for the In Situ Detection of Software Expertise

Tovi Grossman and George Fitzmaurice
Autodesk Research, Canada

Task-based analysis is a common and effective way to measure expertise levels of software users. However, such assessments typically require in-person laboratory studies and inherently require knowledge of the user's task. Today, there is no accepted method for assessing a user's expertise levels outside of a lab, during a user's own home or work environment activities. In this article, we explore the feasibility of software applications automatically inferring a user's expertise levels, based on the user's in situ usage patterns. We outline the potential usage metrics that may be indicative of expertise levels and then perform a study, where we capture such metrics, by installing logging software in the participants' own workplace environments. We then invite those participants into a laboratory study and perform a more traditional task-based assessment of expertise. Our analysis of the study examines if metrics captured in situ, without any task knowledge, can be indicative of user expertise levels. The results show the existence of significant correlations between metrics calculated from in situ usage logs, and task-based user expertise assessments from our laboratory study. We discuss the implications of the results and how future software applications may be able to measure and leverage knowledge of the expertise of its users.

1. INTRODUCTION

A well-recognized property of interactive systems is that its users will have a range of expertise levels. Early studies in human–computer interaction (HCI) have

Tovi Grossman (tovi.grossman@autodesk.com) is a user interface researcher with an interest in software learnability and new input and interaction technologies; he is a Sr. Principal Research Scientist in the User Interface Research Group at Autodesk Research. **George Fitzmaurice** (George.fitzmaurice@autodesk.com) is a user interface researcher with an interest in software learnability and new input and interaction technologies; he is a Director of Research and the Head of the User Interface Research Group at Autodesk Research.

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/hhci.

CONTENTS

told us that it can take a long time before users master the offered functionality of a software system (Nilsen et al., 1993). Typically, a software application has little or no awareness of where the user is along this progression. But understanding a user's expertise could have important implications. For example, help systems could be tailored to meet a user's level of experience (Hurst, Hudson, & Mankoff, 2007; Paris, 1988), information delivery could be personalized (Heckerman & Horvitz, 1998; Teevan, Dumais, & Horvitz, 2005), and interfaces could adapt to provide more advanced features or flexibility to its expert users (Findlater & McGrenere, 2007, 2010; Gajos, Everitt, Tan, Czerwinski, & Weld, 2008; Shneiderman, 2003).

Despite such motivations, little attention has been given to measuring software expertise, in situ, in a user's own home or office environment. Typically, a user's efficiency or expertise with a software application is assessed using a task-based or GOMS analysis (Bhavnani & John, 1998; Nilsen et al., 1993). These forms of analysis capture and compare a user's performance when completing a predetermined task, typically to a model of optimal error-free performance of that task. Such studies are valuable for academic purposes of understanding skill development and how expert and novice users differ. However, this methodology is not practical for the purpose of in situ expertise assessment for two reasons. First, task-based analysis requires the user to perform tasks in a laboratory setting. Second, user performance is evaluated against models that are task dependent. In real-world settings, a software application is not aware of the specific task the user is trying to accomplish, which makes task analysis not possible. For example, the system cannot compare the user's task performance to an error free GOMS model, because the task is unknown.

Initial attempts to automatically detect expertise have been made in recent research literature (Ghazarian & Noorhosseini, 2010; Hurst et al., 2007; Masarakal, 2010). However, such work is limited to tasks where a user can transition from novice to expert within minutes, such as the speed at which users make selections from a menu. Furthermore, those techniques have been validated only in a laboratory setting. An important and open research issue is determining if there are metrics that can be captured from in situ software usage, which can be used to infer the level of expertise of the user.

It is common for this form of high-level software expertise levels to be informally defined along a single dimension, ranging from novice to expert (Dreyfus, Anthanasiou & Dreyfus, 2000; Norman, 1991; Rasmussen, 1987). However, if a software application is ever going to truly adapt to a user's level of expertise, then tagging a user along a unidimensional spectrum may not be sufficient. For example, two users may both be considered intermediate users but may differ greatly in their skills and deficiencies with the software, and thus differ in how the software should adapt to their usage.

Instead of focusing on elemental tasks within a software application, we consider an application-level definition of *software expertise*. Ericsson (2006) defined *expertise* as "the characteristics, skills and knowledge that distinguish experts from novices and less experienced people" (p. 3). Thus, our definition of *software expertise* is as follows:

> ***Definition*** *(Software Expertise).* The characteristics, skills, and knowledge that distinguish experts from novices, considered across the entire scope of functionality that the software provides.

This definition is with respect to a specific software application. Within this definition, the "entire scope of functionality" can refer to a multitude of components of that piece of software, such as individual user interface (UI) elements, executable commands, or higher level workflows that the software supports. As such, this definition can be considered multidimensional in its nature, as a user's skills and knowledge can vary independently from one aspect of the software to another.

In this article, we make two significant contributions to the existing body of research related to detection of software expertise. First, based on a survey of relevant literature, we introduce a multidimensional framework for organizing potential in situ metrics of software expertise. This framework will contribute to our field's understanding of software expertise, and we demonstrate how it can be used to identify new classes of expertise, such as isolated experts, and naïve experts. This leads to our second main contribution, a formal study with AutoCAD, where we evaluate the relevance of these metrics, our ability to measure them in situ, and their correlation with task-based analysis and assessments from two expert judges. A thorough analysis of the data demonstrates a number of interesting patterns in the data and provides evidence that indicators of software expertise levels can be captured in situ. We believe both of these contributions will help develop and improve existing software techniques that measure and leverage the knowledge and skills of users' expertise levels. We conclude by discussion how future software applications may be able to measure and leverage knowledge of the expertise of its users.

2. RELATED WORK

In this section we review the work related to our research efforts of understanding metrics for software expertise. A broad survey on expertise and user modeling in general is beyond the scope of this article. We direct the reader to some of the seminal references on this topic (Anderson, 1983; Anderson & Lebiere, 1998; Ericsson, 2006; Ericsson & Lehmann, 1996; VanLehn, 1989). Next we provide only a brief outline of some of the most relevant work from the cognitive psychology literature. We then discuss the literature that will be most relevant to understanding software expertise metrics from an HCI perspective.

2.1. Cognitive Psychology Theories of Expertise and Skill Acquisition

VanLehn (1996) discussed some of the important differences between novices and experts. In particular, novices focus their attention on solving problems, whereas

experts can carry out necessary tasks based on their existing knowledge. More formally, VanLehn identified three stages involved in cognitive skill acquisition (VanLehn, 1996): (a) the learning of an individual principle; (b) the learning of multiple principles and understanding how to apply them in new combinations; and (c) continued practice, which increases speed and accuracy of performance. This theory of cognitive skill acquisition correlates with the existing frameworks of software learnability (Grossman, Fitzmaurice, & Attar, 2009), where challenges include understanding how to use individual commands, understanding how to use commands in combination to accomplish higher level tasks, and transitioning to more efficient behaviors.

Anderson, Corbett, Koedinger, and Pelletier (1995) provided a model of acquisition of expertise based on the ACT* theory of learning and problem solving (Anderson, 1983). The theory is based on the principle that cognitive skill is based largely on units of goal-related knowledge and that cognitive skill acquisition is derived from the formulation of thousands of these rules and associated task states. Of particular relevance is the distinction between declarative knowledge and procedural knowledge. Declarative knowledge relates to knowing about something and can generally be obtained directly from observation and instruction. Procedural knowledge relates to knowing how to use or do something. Cognitive skill is required to convert declarative knowledge into goal-specific production rules that represent procedural knowledge. Anderson et al. (1995) presented a technique for incorporating such a model of cognitive skill acquisition into an intelligent tutoring system for LISP, geometry, and algebra.

Relating this model to software expertise, a novice user may have declarative knowledge related to knowing about individual commands, whereas a more expert user with advanced cognitive skill would have greater procedural knowledge, manifested as an ability to know how to choose which commands to use to complete higher level workflows appropriately and how to use those commands.

In terms of how expertise is actually obtained, Ericsson, Krampe, and Tesch-Romer (1993) showed that expert performance can be a result of prolonged deliberate practice that is designed to optimize improvement and that individual differences can often be accounted by different practice habits, rather than being a reflection of innate talent. However, Ericsson also provided a nice review of literature (Thorndike, 1921) that has shown "adults perform at a level far from their maximal level even for tasks they frequently carry out" (Ericsson et al., 1993, p. 365) and argued that "the belief that a sufficient amount of experience or practice leads to maximal performance appears incorrect" (Ericsson, Krampe, & Tesch-Romer, 1993, p. 366). The implication to our work is that it cannot be assumed that software expertise will correlate to the amount, or number of years, of experience a user has.

Finally, Ericsson and Williams (2007) discussed some interesting issues related to methodologies for measuring expertise. They argued that "one of the central challenges to studying highly skilled performance in the laboratory is methodological" and that "it is necessary to develop standardized methods that allow investigators to make experts repeatedly reproduce their superior performance in the laboratory"

(p. 115). They discussed the expert-performance approach, where performance is initially captured and elicited in the laboratory using tasks representative of core activities in the domain. Although their work is in the context of expertise in the general domain of cognitive psychology research, there is a relevance to our own work. In particular, there has been little agreement among HCI researchers as to the appropriate metrics and methodologies for measuring expertise, in particular outside of a laboratory setting. In our work, we apply the expert-performance methodology to capture software expertise in a laboratory, and we correlate those measurements to software usage in situ.

2.2. Human–Computer Interaction Research on Expertise

GOMS Models

Some of the fundamental research done in the HCI literature looks at the issue of user expertise. Most notably, research on GOMS modeling decomposes tasks into component activities and considers the expertise level of a user completing those activities (Card, Moran, & Newell, 1983; Gray, John, & Atwood, 1992; John & Kieras, 1996).

A common example is to consider a user who needs to move a phrase in a manuscript using a text editor. This type of example task can be modeled using the KLM variant of GOMS (Card et al., 1983), with a sequence of actions, such as mentally preparing to move the cursor, moving the cursor to the beginning of the phrase, and clicking the mouse button. Guidelines exist for assigning times to each of these component actions and summing them provides a total predicated time for the task.

Typically the models assume error free performance, and in some cases assume that the user has "extreme expertise" (John & Kieras, 1996), for which both mechanical and mental operations are performed optimally. As such, a user's actual performance can be compared to the GOMS model to obtain an approximation of the user's expertise level. We refer the reader to the work of John and Kieras (1996) for a thorough review of GOMS decompositions. The most important issue is that GOMS cannot be used for the detection of expertise in situ, because it requires knowledge of the user's task.

Acquisition of Expertise

An important study on the acquisition of expertise was conducted by Nilsen et al. (1993), where the researchers followed a group of users over a 16-month period. Both quantitative and qualitative aspects of skill development were measured and compared to a benchmark set by expert users. Although the skills that the users were learning were acquired in the user's own workplace, the testing was done in the laboratory, using traditional task-based analysis. Our goal is to identify measurements that could be made outside a laboratory setting.

Particularly relevant to our work is a long line of work by Bhavnani and colleagues that explored expertise in computer-aided drafting and its application to other complex computer systems. Bhavnani classified expertise levels by distinguishing between users with and without *strategic knowledge* (Bhavnani & John, 1996). Users without strategic knowledge tend to rely on more familiar but less optimal strategies. Although such users may be able to execute basic, and sometimes even complex, commands rapidly, they do not exhibit strategies to decompose a task so that unnecessary steps can be avoided (Bhavnani & John, 1996).

Bhavnani also used GOMS decompositions to show how strategic workflows can result in more efficient behaviors (Bhavnani & John, 1998). This decomposition separates a workflow into four layers: a task layer, an intermediate layer, a command layer, and a keystroke layer. This formalization helps reveal why some users do not transition to use strategic knowledge. The cause of this is argued to be the qualitative difference of acquiring the strategic knowledge in the intermediate layer, where the user chooses what combination of commands to use to complete a task (Bhavnani & Bates, 2002; Bhavnani & John, 2000). Bhavnani's distinction of strategic knowledge is similar to Anderson's model of procedural knowledge. This distinction will be incorporated into our framework of software expertise metrics.

Bhavnani observed that efficient strategies involve the use of workflows that aggregate operations and aggregate the elements on which they operate (Bhavnani & John, 1998). Although it was suggested that user behavior could thus be described by the presence or absence of commands that are known to be used for aggregation, this behavior was not investigated by them. Their observations lead us to look at the possibility of classifying expertise on a basis of the types of commands a user works with.

Furthermore, Bhavnani noted that the final quality of a user's work is not necessarily related to a user's level of strategic knowledge, as there may be no visual indication if a drawing had been created with a suboptimal strategy (Bhavnani & John, 1996). However, if someone else begins to work with such a drawing, deficiencies in the workflows may become apparent—for example, if a polygon had been created with multiple individual lines, it could make it difficult to select the entire polygon (Bhavnani & John, 1996). Thus, when judging expertise from the quality of a user's content, the content's visual appearance should not be the only consideration.

Consistent with the cognitive psychology research, Bhavnani observed that even users with formal training and many years of experience may not exhibit strategic knowledge (Bhavnani & John, 1996, 1997). Bhavnani referred to work by Nilsen et al. that found that "skilled novices" could execute commands just as fast as experts but took twice as long to complete tasks because of the methods being used to complete the tasks (Nilsen et al., 1993). Bhavnani developed a specific training course that targeted strategic knowledge and found that this type of training successfully allowed users to recognize opportunities for efficient usage of CAD (Bhavnani, John, & Flemming, 1999). Later, it was shown that teaching of strategic knowledge can be paired with teaching of command knowledge, without harming command knowledge or taking excessive time.

In addition to Bhavnani's work, other research in HCI has described the differences between novices and experts based on their knowledge. Kolodner (1983) stated that "experts are more knowledgeable about their domain and an expert knows how to apply and use his knowledge more effectively than does a novice" (p. 497). LaFrance (1989) provided a set of high level differences between novices and experts, such as "Experts' knowledge is more complex than novices' knowledge" (p. 9).

Measurable Factors of Software Expertise

When considering how expertise should be measured, it is useful to review what specific factors researchers have identified when discussing software expertise. It should be noted that these measureable factors of software expertise do not describe expertise at the application level but may serve as important building blocks. For low-level features, the power law of learning (Card, English, & Burr, 1987) is often used to model progress in expertise, where at some threshold, the user crosses over from novice to expert. Scarr, Cockburn, Gutwin, and Quinn (2011) incorporated such learning curves into a framework of intra- and intermodal expertise development, which expose the factors that affect a user's transition to expert performance.

Nielsen (1994) described a higher level learning curve, applied to overall proficiency with an application. Whereas psychology research has shown that higher level learning and expertise can be decomposed as the learning of individual component skills (Lee & Anderson, 2001), it is not clear that continuous learning curves apply to application-level software expertise. In particular, users often fail to switch to expert interface methods (Scarr et al., 2011).

Usage of UI Components

At the lowest level, expertise has been defined based on usage of individual UI components to access application features. Kurtenbach (1993) defined expert behavior with UI components as "terse, unprompted and efficient actions" (p. 2). Hurst classified this type of expertise as "skill" (Hurst et al., 2007). Scarr et al. introduced several recent techniques that have aimed to support expert performance during command access in GUI applications (Scarr et al., 2011, Scarr, Cockburn, Gutwin, & Bunt, 2012).

Command Efficiency and Vocabulary

A larger amount of work has been aimed at defining expertise based on command or command usage. Bhavnani's framework of expertise has a strong dependence on software commands. An expert, with strategic knowledge, understands how to choose and use commands to efficiently complete tasks, whereas a "skilled novice" may also be efficient with individual commands but not be efficient at completing tasks.

Expertise has also been defined based on a user's command vocabulary. In particular, Linton introduced the skill-o-meter, which compared a user's command

usage patterns to the aggregate of a selected user community (Linton, Joy, Schaefer, & Charron, 2000). McGrenere and Moore (2000) also profiled user's command vocabulary, but their goal was to evaluate the associated software, not the user's expertise levels. Greenberg and Witten (1988) also examined frequencies of command invocations and demonstrated that such frequencies can vary greatly, even within groups of users having similar needs. Pemberton and Robson (2000) conducted a questionnaire and found that only low-level features of spreadsheet software was being commonly utilized, concluding that targeted training is required for users to advance their own expertise levels. Lawson, Baker, Powell, and Foster-Johnson (2009) found that expert users used advanced commands in Microsoft Excel more frequently.

Task Skill

At a higher level, expertise could consider a user's ability to complete tasks with the system. Ghazarian and Noorhosseini (2010) defined *task skill* to indicate the skill level of a user in performing a specific task in an application. Masarakal (2010) defined expertise as "the ability of a user to complete a task" (p. 2). Heckerman and Horvitz (1998) considered expertise to be defined by sets of competency tasks completed successfully. Hurst used the word "expert" to refer to mastery of an activity or task (Hurst et al., 2007).

In a study of spreadsheet software, Lawson compared the spreadsheet practices of beginner and expert users (Lawson et al., 2009). Certain tasks were carried out differently between the two groups. For example, the expert group separated formulas from data input more consistently. As such, task performance should be considered as a metric for software expertise.

In Situ Identification of Expertise

Although the preceding factors may help guide metrics that can be used for expertise, it is still unclear how expertise could be measured in situ.

To identify expertise during actual software usage, Linton compared a user's command usage pattern to a community of users (Linton & Schaefer, 2000). However, the assumption that command usage patterns map to expertise levels has not been validated.

In recent work by Hurst et al. (2007), skilled use of an application was detected based on low-level mouse and menu data. Masarakal (2010) used similar features to classify expertise in Microsoft Word. Ghazarian and Noorhosseini (2010) built automatic skill classifiers for desktop applications. Common among these projects is that they focused on repetitions with specific tasks and UI components, where users could transition from novice to skilled behaviors in the span of minutes. Furthermore, classes of expertise were labeled either based on self-assessment (Masarakal, 2010), or by labeling initial trials as novice behavior, and final trials as expert behavior (Hurst et al., 2007). Thus, this type of work may accurately identify short-term learning

effects, but their validity in terms of higher level learning for complex applications is still unknown.

In the domain of web search behaviors, White, Dumais, and Teevan (2009) characterized the nature of search queries and other search session behaviors by experts and nonexperts. The measurements were captured in situ and allowed the authors to develop a model to predict expertise based on search behaviors. To test their model, experts were separated from nonexperts based on whether they had visited specific predetermined websites (such as the ACM Digital Library for the domain of computer science). This method of separation (using predetermined websites) is not possible for assessing software expertise. As such, we develop a new methodology where we link in situ observations to laboratory expert judgments.

2.3. Summary of Literature Review

In summary, there is a long line of research in cognitive psychology to help us understand the nature of expertise and how it is acquired. Whereas Bhavnani's line of work provides an excellent resource for understanding difference in novice and expert software practices, there are still open questions. In particular, although it is clear that many factors are at play, there is no consensus on what metrics should be used to measure software expertise. Furthermore, although methods to identify expertise exist, we have found no previous attempts to measure expertise in real-world usage settings.

3. SOFTWARE EXPERTISE METRICS

In this section, we provide an organizing framework around metrics for software expertise. The main goal of this framework is to establish the methods for which software expertise can be measured. As presented in our introduction, we establish an application-level definition of software expertise:

> ***Definition*** *(Software Expertise)*. The characteristics, skills, and knowledge that distinguish experts from novices, considered across the entire scope of functionality that the software provides.

To provide formal grounding to this broad definition, we developed a framework of metrics that could be used to assess software expertise, consisting of low-level and high-level metrics (Figure 1). Although our eventual goal is to detect expertise outside of a lab environment, the metrics that our framework encopasses may not all be directly measurable in situ. In our study, we investigate how reliably these metrics can be determined in situ and which, if any, correlate to in-lab assessments of expertise levels.

FIGURE 1. Our framework of software expertise metrics, and examples of how the dimensions of the framework may manifest into behaviors of novice and expert users.

Low-Level Framework Dimensions		Prior Literature	Example User Behavior Manifestations	
			Novice	Expert
UI Expertise	Familiarity (high/low relevance)	-	Is not aware of a contextual menu system	Knows about all of the UI components in the software
	Frequency (high/low relevance)	-	Relies entirely on linear menus to access commands	Frequently uses hotkeys and other UI accelerators
	Efficiency (high/low relevance)	(Kurtenbach, 1993) (Hurst et al., 2007)	Is inefficient when using the ribbon	Accesses menu items equal to the expert performance time predicted by GOMS
Command Expertise	Familiarity (high/low relevance)	(McGrenere & Moore, 2000) (Greenberg & Witten, 1988)	Is only aware of the system's basic functionality	Knows about all of the advanced commands
	Frequency (high/low relevance)	(Linton & Schaefer, 2000) (Matejka et al., 2009) (Li et al., 2011)	Advanced commands only make up 5% or less of total command usage	Uses a combination of basic and advanced commands in all work sessions
	Efficiency (high/low relevance)	(Bhavnani & John, 1996)	Inefficient when applying individual commands	Efficiently uses individual commands
Task Expertise	Familiarity (high/low relevance)	-	Is only aware of the basic tasks that can be completed.	Understands capabilities and limitations of the software
	Frequency (high/low relevance)	(Lawson et al., 2009)	Rarely performs advanced tasks	Often performs advanced tasks
	Efficiency (high/low relevance)	(Bhavnani & John, 1996) (Ghazarian & Noorhosseini, 2010) (Masarakal, 2010)	Has many pauses while completing high-level tasks	Can complete a given sequence of operations efficiently
High-Level Framework Dimensions				
Appropriateness of Workflows		(Bhavnani & John, 1996)	Chooses inefficient workflows to complete tasks	Chooses advanced and efficient ways to complete tasks
Quality of Content		(Bhavnani & John, 1996)	Underlying structure of content is problematic	Creates high quality and reusable content
Domain Knowledge		(Kolodner, 1983) (Nielsen, 1994)	Lacks knowledge related to domain	Very knowledgeable about domain

3.1. Low-Level Metrics

A common theme among the psychology literature we reviewed is that the building blocks for high-level expertise can be based on core individual component skills (Lee & Anderson, 2001). For this reason, we include low-level metrics of software expertise, representing the building block component skills. We distinguish between such component skills by establishing a *Scope* dimension. In addition, we introduce a *Measurement Type* dimension, which considers how a user's skill within any of these scopes could be measured. Last, a *Component Relevance* dimension distinguishes between components of the software that are relevant and irrelevant to a user with respect to their usage domain.

The top section of Figure 1 illustrates our framework of low-level software expertise metrics, across the three dimensions. This framework combines the three dimensions to generate 18 unique metrics of software expertise. For example, one

metric within the *UI Expertise Scope* would be "Familiarity with highly relevant UI components."

Scope

The first low-level dimension of the framework is the *Scope*, which includes *UI Expertise*, *Command Expertise*, and *Task Expertise*.

UI Expertise. In this scope, we consider the user's expertise level with operating the user interface components (Hurst et al., 2007; Kurtenbach, 1993), such as menu systems, independent of the commands to which they provide access. For example, in an application with a ribbon, we may consider the user's expertise with using the ribbon mechanism, such as how quickly they locate and click on the bold icon. For an application that used marking menus (Kurtenbach, 1993), we would consider the user's expertise with the marking menu, such as how quickly they select second level items from the menu.

Command Expertise. At the command level, we consider the user's expertise levels with individual features of the system (Bhavnani & John, 1996; Linton & Schaefer, 2000). Such features may be referred to as *commands* or *tools*. For consistency, in this article we refer to individual features as *commands*.

For example, in an image editing application, we may choose to consider how many blur commands the user knows, or how efficient the user is with individual drawing commands. In some applications, there are commands that can be executed by clicking a single button in the UI, such as the bold command in a document editor. In such cases, command expertise could be based on a user's awareness of the command or how often they use it. The actual selection of the command from the interface would relate to UI expertise. In other cases, individual commands can have complex usages, such as a gradient command in an image editor. For such commands, Command Expertise would additionally relate to how efficiently the user works with the command.

Task Expertise. We also want to consider a user's expertise when performing actual tasks (Bhavnani & John, 1996; Ghazarian & Noorhosseini, 2010; Heckerman & Horvitz, 1998; Hurst et al., 2007; Masarakal, 2010). A task can be defined as a target goal that a user wishes to accomplish. Typically the goal is to achieve a desired effect on the application content or environment, and the user must choose an appropriate *method* (or workflow) to accomplish this *goal*, made up of a sequence of actions (Card et al., 1983). *Task Expertise* refers to a user's ability to carry out a selected method to achieve a goal. In particular, a user who can efficiently complete a given sequence of actions efficiently would be demonstrating *Task Expertise*. For example, their performance level may be close to the time predicted by a GOMS decomposition of the chosen sequence of actions.

Measurement Type

To better understand each of the aforementioned scopes of expertise, we need to consider how each scope is measured. The *measurement* subdimension is applied to each of the previously described scopes and provides a framework for how expertise at each of these levels can be assessed. The levels of this subdimension include *Familiarity*, *Frequency*, and *Efficiency*.

Familiarity. This metric considers what level of knowledge a user has. For *Command Expertise*, one may consider an expert someone who knows about every command available in an application, regardless of how often he or she uses it (McGrenere & Moore, 2000). For *Task Expertise*, this provides a metric of a user's awareness of the limits and capabilities of a software application.

Frequency. This metric considers the frequency of actual usage. Command frequency has been used previously to profile user expertise (Li, Matejka, Grossman, Konstan, & Fitzmaurice, 2011; Linton & Schaefer, 2000; Matejka, Li, Grossman, & Fitzmaurice, 2009; Rooke, Grossman, & Fitzmaurice, 2011). For *UI Expertise*, a higher frequency of usage with modifier hotkeys may be a sign of a higher level of expertise (Grossman, Dragicevic, & Balakrishnan, 2007). Or, for *Task Expertise*, a user who more frequently performs advanced tasks, such as dividing spreadsheets into integrated modules (Lawson et al., 2009), may be considered to have higher expertise.

Efficiency. The efficiency metric considers the user's actual performance levels. Typically, this would mean how quickly the user interacts with the system—whether it be interacting with UI components (Hurst et al., 2007; Kurtenbach, 1993), executing individual commands (Bhavnani & John, 1996), or completing entire tasks (Bhavnani & John, 1996; John & Kieras, 1996). For *Command Expertise*, efficiency is only a relevant measure for the commands that have nontrivial usages, beyond selecting them from the UI; the act of selecting such commands from the UI would fall within the category of *UI Expertise*.

Component Relevance

Research on complex applications has shown that any individual user may only use a subset of a software application's commands (McGrenere & Moore, 2000), and some commands have higher relevance to users than others (Li et al., 2011; Matejka et al., 2009). We define *component relevance* as the level of importance that a component of a software application has to a user, based to that user's day-to-day tasks. For example, the 3D tools in a drafting application have a high level of relevance to a 3D modeler but a low level of relevance to someone who creates only 2D drafts. As such, if a user is observed as having poor performance with these 3D tools, it might be important to first consider how relevant those tools are to the user before assessing the user's expertise level.

3.2. High-Level Dimensions

Although the low-level dimensions are important to consider, a user's true expertise, at the application level, is more than the sum of these parts. From our survey of related work, we identified three important high-level dimensions that relate to a holistic consideration of software expertise (Figure 1, bottom). It is interesting to note that these high-level metrics may not be as easily quantified or directly measured as the low-level metrics described previously. In Section 5 we study potential correlations between the low-level metrics and high-level assessments of software expertise.

Appropriateness of Workflows

We define a workflow as a method, or sequence of operations, to complete a goal task. Previous research has shown that a distinguishing factor of expert users is that they have the ability to choose appropriate workflows (or methods) to accomplish their tasks (Bhavnani & John, 1996; Kolodner, 1983; Rasmussen, 1987).

It is important to note that *Appropriateness of Workflows* is distinct from *Task Expertise*. For example, consider a user who needs to add five rows to a table in a document editor. The user may choose to do this by adding one row at a time, and perform the sequence of operations to do so very efficiently. However, there may have been an alternative workflow, or sequence of operations, that would have been a more appropriate. The user, in this example, would be demonstrating *task expertise* but not higher level strategic knowledge or *Appropriateness of Workflows*.

Quality of Content

Another important factor to consider, when assessing software expertise, is the actual quality of the work produced when using the system. It would certainly be hard to label a user as an expert, if that user could not produce high-quality results. However, in some cases, such requirements may not be placed on an "expert." For example, an instructor may have the skills to teach students about the software but may not have adequate "Contributory Expertise" (Collins & Evans, 2002) to produce high-quality results with using the software.

Bhavnani (Bhavnani & John, 1996) has noted that content created with an inefficient workflow may be visually indistinguishable from content created more efficiently by a user with strategic knowledge. As such, when considering content quality, a deeper understanding of quality, beyond the content's surface appearance, should be considered.

Domain Knowledge

Our final consideration of software expertise is the domain knowledge possessed by the user. Domain knowledge has been previously identified as an important aspect to software learning, and as such should also be considered when discussing expertise (Cote-Munoz, 1993; Grossman et al., 2009; Nielsen, 1994). In some cases, the quality

of work may be dependent on domain knowledge. For example, in a sketching application, a user may have strong expertise with the software application but may create poor content because of a lack of the associated domain knowledge and skill of artistic sketching (Fernquist, Grossman, & Fitzmaurice, 2011).

3.3. Summary

Figure 1 provides a summary of the related literature that led to the derivation of each of the dimensions we have discussed. It is interesting to note that researchers have previously discussed almost all combinations of our multidimensional framework, with the exception of Familiarity and Frequency within UI Expertise, and Familiarity within Task Expertise. This exposes new potential ways to consider software expertise and could potentially allow the generation of new metrics related to expertise.

Figure 1 also provides examples of how each dimension of the framework manifests into user behaviors, both at a novice and expert level. These examples may help designers identify techniques to help users shift from novice to expert behaviors, specific to each form of expertise.

For brevity, the *Component Relevance* dimension is aggregated within each metric of the table, for the example user behavior manifestations. In essence, each of the examples provided could be within the scope of components that are either relevant or not relevant to the user.

It can also be noted that some of the dimensions of our framework may interact with one another. For example, a user with high command expertise with a certain set of tools may be more likely to also exhibit a high level of UI expertise for accessing that set of tools. Alternatively, it may be the case that a user is efficient when using a tool but uses inefficient UI mechanisms to access that tool.

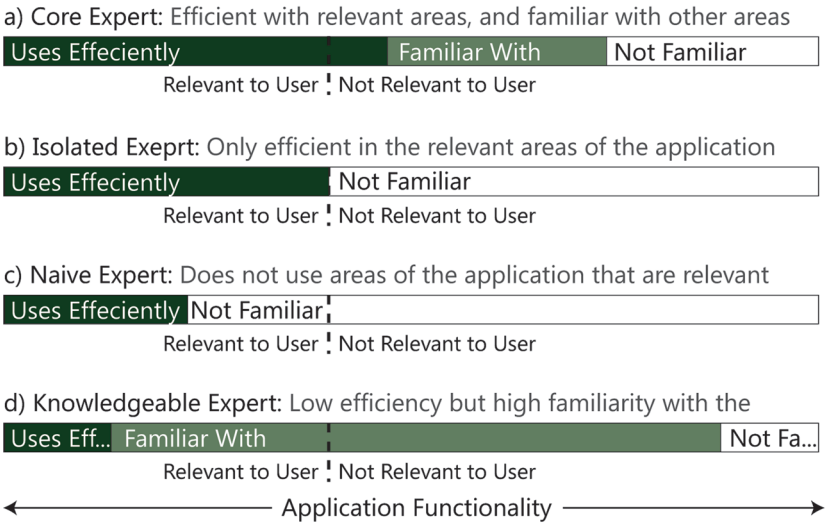
4. CLASSIFICATION AND MEASUREMENT OF SOFTWARE EXPERTISE

4.1. Classes of Expertise

A main contribution of the aforementioned framework is that it organizes possible metrics of software expertise into a coherent structure. This can serve as a reference for researchers or practitioners when attempting to measure the expertise levels of software users. An additional contribution of the framework is that it can serve as a tool to provide new classifications of software expertise levels, beyond the traditional unidimensional spectrum ranging from novice to expert.

By just considering the low level metrics of the framework, we obtain a multidimensional definition of expertise based on the familiarity, frequency, efficiency, and relevance of the entire application space (Figure 2). Next we sample some possible categories of expertise that can be represented using this framework.

FIGURE 2. A conceptual visualization of how low-level metrics can result in different expertise profiles.



***Note*. The *x*-axis represents the entire space of the application functionality. This space is divided into areas that are and are not relevant to the user, and also divided into spaces that the user uses efficiently, is familiar with, and is not familiar with.**

Core Expert. Findlater and McGrenere (2010) described core task performance as performance of completing known or routine tasks. Using our framework, we could define a *core expert* as someone who demonstrates efficient performance with the commands and tasks that are most relevant to them but may also exhibit a high level of performance with functionality of the application that aren't necessary for their day-to-day tasks. Users could have good performance with less relevant functionality if they have a high level of knowledge of how the application works, prior experience, or a deep understanding of the interaction metaphors of the application.

Isolated Expert. An isolated expert is only familiar with the limited area of the software that is relevant to their domain of usage. Such a user may demonstrate expertise within this scope of the application space but has little knowledge or usage experience in other areas. Unlike a core expert, who may have enough knowledge to complete tasks outside their main usage domain, an isolated expert would likely struggle with tasks outside of their normal usage patterns.

Naïve Expert. Previous research has shown that even experienced users may be unaware of areas of an application that are relevant to their usage domain (Grossman et al., 2009). We define a naïve expert as a user who has high efficiency with the commands that he or she uses frequently but is unaware of other commands

that are relevant to the work (Figure 2c). This is in contrast to core experts and isolated experts, who are familiar with the majority of commands relevant to their work. Naive experts would be good targets for learning aids that promote feature awareness, such as command recommenders (Linton & Schaefer, 2000; Matejka et al., 2009).

Knowledgeable Expert. A user who has high familiarity at each level of *Expertise Scope* regardless of his or her ability to actually use the system could be classified as a knowledgeable expert (Figure 2d). This is directly related to Collins and Evans's (2002) distinction of *Interactional Expertise*, which is defined as enough expertise to understand and interact within the domain but not necessarily enough to contribute to the domain. Within the realm of software, this type of profile may be possessed by product support specialists, or technical writers that document the system.

4.2. Methodologies for Assessing Expertise

One of our motivations for developing a framework of expertise is to eventually enable software applications to identify the expertise levels of its users. Earlier we defined various classes of software expertise that could be identified. But there is still the problem of *how* such classes of expertise can be identified. From our review of related work, we have identified four potential methodologies for determining software expertise at the application level, which are described next.

Self-Assessment. Users rank their own expertise through questionnaires or dialogs in an application (Masarakal, 2010). This method is commonly used in usability studies to correlate the primary results of the study to user expertise levels. This method has an advantage that it is easy to perform and can be done outside of a laboratory, but it may not be reliable (Nisbett & Wilson, 1977).

Expert Assessment. A potentially more reliable method to assess expertise levels is to have one or more judges observe and rate a user's level of expertise (Einhorn, 1974). It would be assumed that such judges would require expertise in the software themselves, particularly in the high-level dimensions of our framework. The drawback of this technique is that it is impractical to perform outside of a lab.

Laboratory Tasks. With this methodology, user's perform controlled tasks in a laboratory setting, and metrics from those tasks are collected and analyzed (Ghazarian & Noorhosseini, 2010; Hurst et al., 2007; Masarakal, 2010). This method has already been shown to be reliable for shortterm learning effects and could potentially identify higher level expertise. As with the expert assessment, the drawback is that the methodology cannot be used in situ.

In Situ Measurements. The collection and analysis of usage metrics from search engines and web browsers are commonly used to understand how people use online systems (Adar, Teevan, & Dumais, 2008; e.g., Teevan, Dumais, & Liebling, 2008, 2010). Numerous recent research projects have also looked at capturing usage logs during software application usage (Akers, Simpson, Jeffries, & Winograd, 2009; Lafreniere, Bunt, Whissell, Clarke, & Terry, 2010; Li et al., 2011; Linton & Schaefer, 2000) and lower level input activities (Chapuis, Blanch, & Beaudouin-Lafon, 2007; Evans & Wobbrock, 2012; Gajos, Reinecke, & Herrmann, 2012; Hurst, Mankoff, & Hudson, 2008). These previous projects have discussed the benefits and challenges of capturing data from in situ usage logs. In particular, it may be possible that some of the metrics traditionally captured from laboratory tasks could be inferred from these usage metrics—for example, the time required to access menu items. This approach has a significant advantage that it does not require any action on the part of the user, aside from logistical considerations such as installing plug-ins or uploading data logs. However, it is unknown how reliable such metrics would be for indicating expertise levels. Furthermore, applications would need to be instrumented to collect the relevant data.

5. USER STUDY

We have presented a survey of research in software expertise, which revealed a diverse range of ways in which software expertise can be defined and measured. We then produced a framework, which organizes previously and newly defined metrics of expertise into a coherent framework. Given an eventual goal of identifying expertise levels externally (without lab studies or expert assessments), two main questions arise from the discussion in these previous sections.

Q1: How well can each of the metrics from the framework be identified? In Section 4.2 we summarize four different methodologies, but we would like to establish which metrics each of these methods can capture. In particular, showing what can and cannot be inferred from these methodologies will be valuable in trying to advance the existing research on identifying expertise levels in-situ.

Q2: Which aspects of the framework correlate to an expert judge's assessment of software expertise? Researchers have already shown the ability to identify metrics related to expertise at certain levels of our framework. For example, Hurst's identification was based on the efficiency metric at the UI Expertise scope (Hurst et al., 2007). But it is currently unclear how indicative this is of software expertise at the application level. Understanding this issue will have important implications for identifying software expertise.

Our study differs from prior art in three ways. First, we investigate the issues of automatically identifying software expertise across the scope of the entire application, whereas previous work has looked at the short-term effects of learning based on repeating similar tasks. Second, we correlate our measurements in the lab with actual

usage logs from home and workplace usage. Third, we compare our collected metrics to assessments made by *expert judges*, to provide better insights and grounding to the results.

5.1. Methodology

Target Application

Our evaluation is carried out with AutoCAD, a computer-aided design software application most commonly used in the architecture industry. Historically, AutoCAD was designed to allow designers and architects to create their 2D drafts and blueprints within a digital medium. The software provides a graphical UI for creating, scaling, and annotating such drawings. All drawings are built up using a series of commands, with the availability of more than 1,000 commands to choose from. For example, a user may use the *LINE* command to draw the outline of a building and then use the *OFFSET* command to create a copy of that outline, offset by a distance representing the thickness of the walls, and then use the *TRIM* command to remove any intersecting edges. Almost all workflows consist of such series of command operations. Most commands are modal; to trim a line, the user selects the trim command, then performs a series of steps to trim the line, and then completes the command by pressing the Enter key.

More recently, AutoCAD has introduced 3D commands for designing 3D objects and models. Overall, the 3D commands are not used as commonly, as there are other dedicated software packages for 3D modeling. However some expert users do use these 3D commands. For example, a 2D design may be extruded into 3D to get a sense of its true volume. The 3D tasks are typically accomplished using workflows consisting of a series of 3D commands.

Overall, AutoCAD serves our purposes well, as it is a complex application that can take years to achieve true expertise and has numerous domains of usage. As such, it has previously been used as a software domain for studying software learnability and expertise (Bhavnani & John, 1996; Grossman et al., 2009; Lang, Eberts, Gabel, & Barash, 1991).

Participants

Sixteen participants were recruited by advertising at local architecture firms, local universities with architecture programs, and online classified websites. A recruitment questionnaire surveyed potential participants on their education level, and level of experience with AutoCAD, both as a student and professionally. Participants were selected to obtain a spectrum of experience levels.

Two participants withdrew prior to completing the laboratory phase of the study. Our analysis is based on the 14 remaining participants (six female, eight male). These participants ranged in age from 22 to 58, with AutoCAD usage experience ranging from 6 months to 10 years. Participants were paid $150 upon completion of the study.

Expert Judges

In addition, we recruited two experienced AutoCAD users to serve as *expert judges* to provide their own assessments of the participants' expertise levels during the study. The first expert was an internal employee, but also a professional architect with 14 years of experience using AutoCAD, and had taught AutoCAD at the undergraduate level for 4 years. The second expert was also a professional architect, with 20 years of experience using AutoCAD and 1 year of experience teaching AutoCAD at the undergraduate and graduate levels. This expert was external and was paid $400 for his participation. Although the expert judge assessments may not necessarily provide a ground truth (Einhorn, 1974), they should be able to provide a holistic measure to which other metrics can be compared.

Setup

The study took place across four group sessions lasting 2.5 hr, with each participant attending one of those sessions. There were two sessions of three participants and two sessions with four participants. We chose to run the sessions in groups in consideration of the length of the study and the required time commitment of the *expert judges*. Each group session consisted of six phases, which are described next.

Each user was provided with a laptop and USB mouse. The sessions took place in a conference room, with the participants seated around a large conference table. Participants were spaced so that they could not see one another's screens and were asked not to interact with one another. An experimenter was present throughout the entire study. Video screen capture software was run on each laptop, and audio was recorded with the laptops' microphones.

Study Phases

Next we describe the six phases of each group session. In each heading, we highlight which of the four methodologies from Section 4.2 that each phase corresponds to. The first phase was conducted in situ, whereas the remaining phases took place in our laboratory.

Phase 1: Usage Data Collection (In Situ Measurements Methodology)

In this phase, usage data statistics were captured for each of the study participants in their natural home and working environments. All participants installed a custom-written AutoCAD plug-in 2 to 4 weeks prior to the laboratory session. The plug-in runs in the background and collects the usage data. Participants were not given any particular tasks to complete during this phase; they continued to use AutoCAD as they normally would. However, participants were asked to use AutoCAD for at least 8 hr prior to the laboratory session, to collect sufficient usage data. The plug-in collected

the following information:

- The name of all commands executed.
- The start and end times of all commands.
- The UI method used to access the command (shortcut, menu/ribbon, command line).
- The time of all mouse and keyboard events.

The usage logs were stored locally, and participants submitted them at the end of the study.

Phase 2: Questionnaire (Self-Assessment Methodology)

Upon arriving to our laboratory, participants completed a short questionnaire, which had questions about their usage experience with AutoCAD, and relevant training or education. Participants ranked their own expertise level on a 5-point scale.

To capture *familiarity* and *frequency* levels at the *UI* scope, the survey also asked participants to rate their familiarity and usage levels with four main UI access points in AutoCAD—the ribbon, menu, command line, and shortcuts, using a single question each, such as *Please rank your familiarity with the AutoCAD Command Line.*

Phase 3: Command Familiarity Survey (Self-Assessment Methodology)

This phase was a test used to capture *familiarity* and *frequency* levels at the *command* scope. Users ran a windows application that surveyed them on 500 AutoCAD commands. The 500 commands were chosen by popularity, calculated as the percentage of users that use the command, as reported through Autodesk's Customer Involvement Program (http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=12264110), using logs from slightly fewer than 3,000 users. Using radio buttons, users were asked to report their familiarity with the command (unfamiliar, familiar), and their frequency of usage (never, infrequent, frequent). Participants were told to only respond as being familiar if they knew the command existed and knew what it is used for. This phase took approximately 30 min.

Phase 4: UI Access (Laboratory Tasks Methodology)

This phase was an experiment used to capture *efficiency* levels at the *UI* scope, and in particular with AutoCAD's two main menu systems—traditional drop-down menus and the ribbon. The experiment consisted of 96 trials and was run using a custom plug-in within AutoCAD. In each trial of the experiment, users were presented with a command and its associated icon. After clicking a "go" button, users had to click on the command in either the ribbon or menu as fast as possible. The experiment was divided into two conditions (*menu, ribbon*), with each condition having 16 commands, repeated in random order across three blocks. The 16 commands contained a mix of eight common (popularity $> 50\%$) and eight uncommon (popularity $< 10\%$)

commands. If the user could not find the command after 30 s, the experimenter would help. This phase took approximately 25 min.

Phase 5: Command Tasks (Laboratory Tasks Methodology)

This phase was an experiment to capture *efficiency* levels at the *command* scope. In this phase, the users were tested on tasks that only required the usage of a single command. The phase consisted of eight trials, in the following fixed order: two common 2D drafting commands, two uncommon 2D drafting commands, two common 3D commands, and two uncommon 3D commands. Our hope was that 2D and 3D would approximate relevant and irrelevant for most users, and the common and uncommon commands would closely approximate familiar and unfamiliar commands for most users. If the user did not complete a task in 5 min, they proceeded to the next task.

Phase 6: High-Level Tasks (Expert Assessment Methodology)

This phase was used to capture *task* metrics and *higher level requirement* metrics. The two expert judges worked together to design four 10- to 15-min tasks. The tasks were a simple 2D drafting task, an advanced 2D drafting task, a simple 3D modeling task, and an advanced 3D modeling task. A formal GOMS decomposition was not conducted, because depending on the strategy chosen, each task would consist of several hundred or even thousands of steps at the individual keystroke level and a sequence of approximately 50 to 100 command executions. As in the command tasks phase, these tasks were meant to elicit a rough approximation of combinations of relevant–irrelevant, and familiar–unfamiliar tasks.

During this phase, the expert judges monitored the user's behaviors and task progress by walking around the room and recording observations. Both judges had extensive experience assessing user expertise levels with AutoCAD, from their classroom and teaching experience. In particular, the judges provided a holistic assessment of the expertise levels which they observed. They were not only rating efficiency but judging the strategies participants chose, the knowledge which they demonstrated, the final quality of their work, and the overall fluidity of the workflows. In general, the nature of these observations could not be easily operationalized and measured computationally.

Each task lasted approximately 10 to 15 min, with the total phase lasting 45 to 60 min. This ensured that the experts had time to get a fair assessment of each of the users in the group. For each of the four tasks, the expert judges were given worksheets on which they could record notes for each participant. After each of the four tasks, the experts assessed user expertise for that task on a 5-point Likert scale. After the entire phase was completed, the judges also provided an *overall expert assessment* for each participant, based on their 45 to 60 min of observations.

The expert judges monitored and rated participants independently and did not confer with one another. In addition, this was the only phase that the experts were present for, to ensure their judgments were not biased by previous phases.

5.2. Results

Laboratory and Assessment Results

To analyze our results, we first profile our participants by analyzing their questionnaire responses (*Phase 2*) and the assessments made by our expert judges (*Phase 6*).

Participant Overview

During recruiting, we tried to obtain a pool of participants that would represent a spectrum of expertise levels, roughly estimated by their profession, and their amount of experience. Figure 3 shows an overview of the participants' experience and self-assessed expertise, as well as the judges overall expert assessment of the expertise levels. The expert assessments were averaged between the two experts. We also calculated a 2D expert assessment by averaging the first two task assessments and a 3D expert assessment by averaging the last two task assessments. To test for interjudge reliability, we performed a correlation analysis across each of the scores for the four tasks and overall assessments, provided by the judges, which showed considerable agreement (Spearman's $\rho = .765$, $p < .0001$).

For ease of presentation, we have sorted the participant numbers by their overall expert assessment score. Overall, it can be seen that this pool of participants represents a nice spectrum of experience and expertise levels. We believe that this diversity is a reasonable approximation to the actual population of AutoCAD users, which also greatly varies. Although exact statistics are not readily available, Autodesk has approximately 12 million professional users and 6.5 million users from the education community (Shaughnessy, 2013). A large proportion of that education community would be novice students learning to use the software.

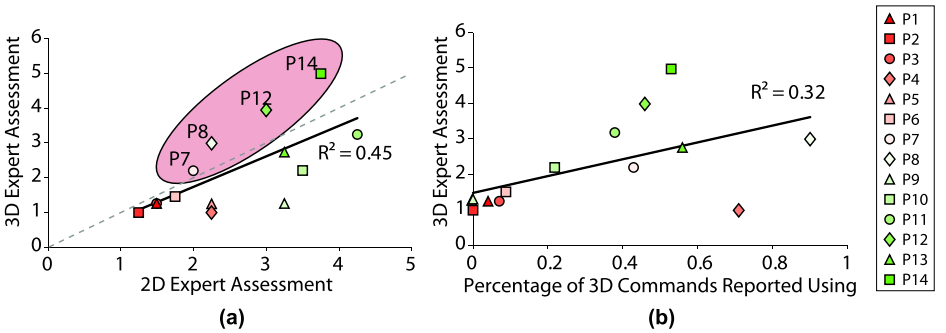
Based on the overall expert assessments, we would say that participants P12, P13, and P14 had the highest level of software expertise and participants P1, P2,

FIGURE 3. Summary of the study participants.

	Self-Assessed Expertise	Experience (Years)	2D Expert Assessment	3D Expert Assessment	Overall Expert Assessment
P1	1	0.5	1.5	1.25	1
P2	3	10	1.25	1	1.5
P3	4	1	1.5	1.25	1.5
P4	4	6	2.25	1	2
P5	4	6	2.25	1.25	2
P6	2	5	1.75	1.5	2.5
P7	4	4	2	2.25	2.5
P8	3	5	2.25	3	2.5
P9	4	10	3.25	1.25	3
P10	4	6.5	3.5	2.25	3
P11	3	3	4.25	3.25	3
P12	3	10	3	4	3.5
P13	3	10	3.25	2.75	3.5
P14	3	6	3.75	5	4

Note. **Participant numbers have been sorted by overall expert assessment.**

FIGURE 4. a) Comparing users' 2D and 3D expert assessments. Four users (highlighted) had 3D assessments that surpassed their 2D assessments (above the dashed line) (*N* = 14). b) Comparing users' 3D command usage and 3D expert assessment (*N* = 14).



P3 and P4, had the lowest, with the remaining participants falling somewhere in between. However, as discussed in the previous sections, software expertise cannot just be considered along a single dimension, which we look at more thoroughly in the remaining analysis.

Some interesting observations can already be made. First, the 2D expert assessments did have a positive correlation with the 3D expert assessments ($r^2 = .45$, $p < .01$; Figure 4a). The four users who are highlighted in this figure are discussed in the next section. This correlation indicates that users who are strong in one area of an application will be more likely to show strengths in another area. It should be noted that the judges knew which participants they were assessing, so this effect may partially be due to a carry-over effect of the assessments.

However, neither the user's years of experience nor the user's self-assessed expertise had any observable correlation with our expert's 2D, 3D, and overall assessments (Figure 5). *This suggests that both years of experience and self-assessment of expertise levels may be misleading indicators of expertise.*

Command Familiarity and Frequency of Usage (Phase 3)

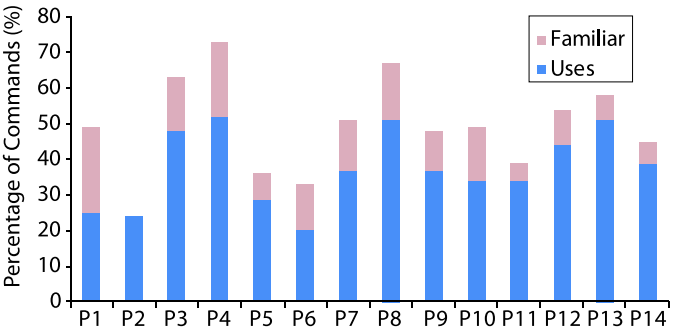
Here we discuss users' self-reported familiarity and frequency usage of AutoCAD commands. Figure 6 demonstrates that users have a spectrum of familiarity and

FIGURE 5. Correlation values of the study participant data summarized in Figure 3 (*N* = 14).

Measure	Experience (Years)	2D Expert Assessment	3D Expert Assessment	Overall Expert Assessment
Self-Assessed Expertise	$R^2 = 0.07$ (p = 0.37)	$R^2 = 0.05$ (p = 0.46)	$R^2 = 0.01$ (p = 0.74)	$R^2 = 0.04$ (p = 0.52)
Experience (Years)	-	0.08 (p = 0.34)	0.02 (p = 0.62)	$R^2 = 0.25$* (p = 0.07)
2D Expert Assessment	-	-	**$R^2 = 0.45$* (p = 0.009)**	**$R^2 = 0.70$** (p = 0.0002)**
3D Expert Assessment	-	-	-	**$R^2 = 0.65$** (p = 0.0005)**

** p < 0.05 | * p < 0.10

FIGURE 6. Percentage of commands familiar with and used ($N = 14$).



usage levels—it also shows that only one user (P4) worked with more than half of the commands in the system. This is consistent with prior research studying software command usage (Greenberg & Witten, 1988; McGrenere & Moore, 2000).

It is interesting to note that P2, who has 10 years' experience, did not report familiarity with any command that he hadn't used. This may be due to P2 being reluctant to report familiarity, but it also indicates that P2 may fall under our classification of an *isolated expert*.

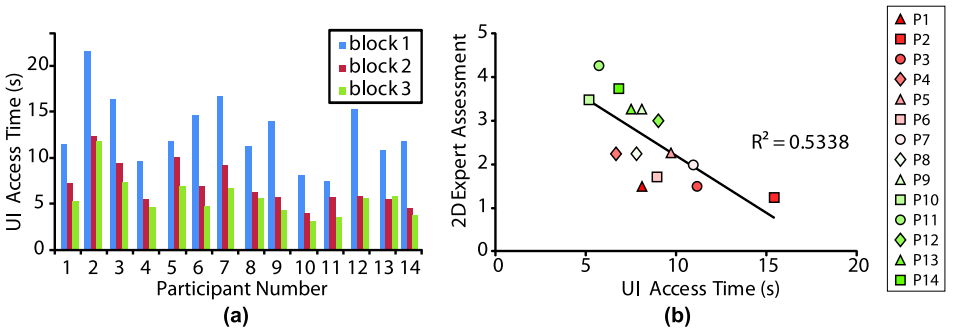
An interesting result is that when comparing these spectra of usage and familiarity to the expertise assessments, we did not find any correlation between the number of commands users use or are familiar with and the overall expert assessments. *This is evidence that a strong assessment of expertise does not necessarily require the usage or familiarity with more commands.*

We can also use this analysis to revisit the data from Figure 4a, which showed some users' 3D assessments surpassed their 2D assessments (P7, P8, P12, P14). As it turns out, all four of these users were familiar 3D users, reporting to use more than 40% of the 3D commands in AutoCAD. In general, there was a correlation between 3D usage and 3D assessment ($r^2 = .32$, $p < .05$; Figure 4b). *This shows that although command familiarity and usage may not be consistent with an expert's assessment of a user's overall expertise level, it may indicate areas of expertise in certain core areas.*

UI Access (Phase 4)

Next we consider the UI access times from *Phase 4* of our study. As expected, block had a significant effect on acquisition times, $F(2, 26) = 135.2$, $p < .0001$. Here we see an interesting effect—for all participants, the completion times decreased between blocks, most prominently between the first and second block (Figure 7a). This demonstrates that even a novice user could exhibit skilled behavior if accessing an item that they had recently accessed. The same may hold true for items the user frequently accesses. Despite this decrease of time, overall UI access time across all blocks did exhibit a significant correlation with users' 2D expert assessments ($r^2 = .53$, $p < .005$; Figure 7b).

FIGURE 7. a) User interface (UI) access completion times, by block. The completion times decreased between blocks, most prominently between the first and second block. b) Correlation between UI access times and 2D expert assessment ($N = 14$).



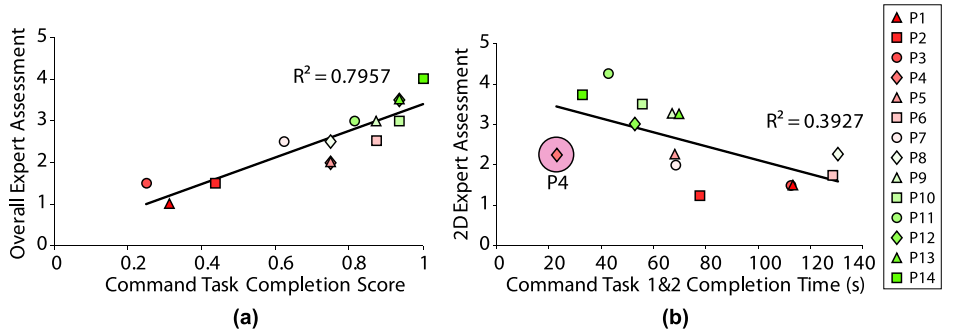
Command Tasks (Phase 5)

Users were assigned a completion score in the command tasks, based on whether a trial was completed (1), almost completed (0.5), or not completed (0). We also recorded task completion time. Overall task completion score had a correlation to the user's overall expert assessment ($r^2 = .79$, $p < .0001$; Figure 8a).

It is interesting to note is that the correlation level is also significant ($r^2 = .75$, $p < .0001$) when only considering the four tasks that were chosen to be previously unfamiliar to users. We verified the tasks were indeed unfamiliar by using the data from Phase 3 of the study. As such, this finding is consistent with the cognitive psychology literature stating that experts are able to use procedural knowledge to adapt and apply their skills in new ways (Anderson et al., 1995).

For completion times, we considered only Tasks 1 and 2, as completion rates were low for Tasks 3 to 8 (the completion rate was 92.9% for Tasks 1 and 2 and 58.3% for the remaining tasks). P1 and P3 did not complete Task 2, so we assigned completion times of 300 s for these tasks. The correlation between completion time

FIGURE 8. a) Correlation between command task completion score and overall expert assessment ($N = 14$). b) Correlation between command task completion time and 2D expert assessment (tasks 1 & 2 only) ($N = 14$).



and 2D expert assessment was significant ($r^2 = .39$, $p < .05$). It is interesting to notice that P4 (Figure 8b, highlighted) was the fastest overall but had a low 2D expert assessment. Thus, P4 would fall under our category of a *naive expert*—very efficient with certain commands but lacking the knowledge to be able to demonstrate expertise when completing the higher level tasks.

It is important to recall that the experts were not present during *Phase 5*, so the command usage metrics are independent from the expert assessments. *This demonstrates that command efficiency is a possible metric for software expertise.*

In Situ Metrics

We now look at the data captured from the usage logs. This is an especially interesting part of the analysis, given that previous work has not focused on the relation between in-situ usage data logs to software expertise. Figure 9 summarizes the metrics calculated from the log files. In the following sections we discuss why each of these metrics was chosen and what they could imply.

Sessions are defined as instances when the AutoCAD application was launched. To calculate active usage hours, we subtracted any time interval longer than 5 min without an input event being recorded. Four users (P4, P7, P8, P11) had less than 6 hr of active usage hours logged. Their data have been omitted from the following *Command Distribution* and *Menu Access* analysis, because it was not complete enough to accurately calculate these metrics.

Command Distribution

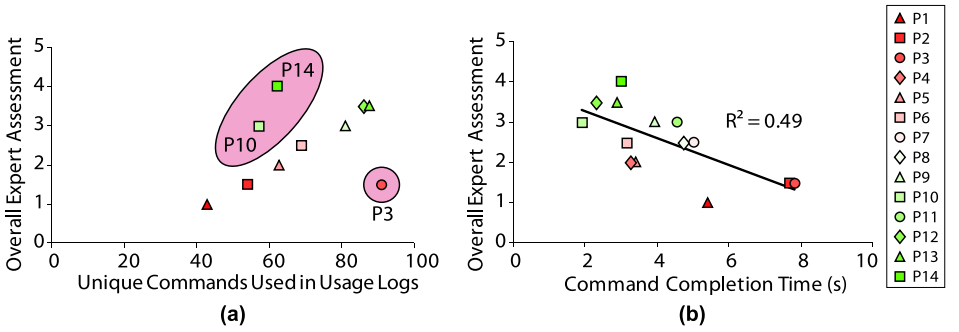
We first look at the nature of the commands users executed, as the type of command a user uses may indicate expertise level (Lawson et al., 2009; Linton & Schaefer, 2000). We did not find any correlation between the unique command counts with the expert assessments. We also looked at the average popularity of the commands used, thinking that expert users possibly use more "rare" commands.

FIGURE 9. Summary of Usage Log Statistics ($N = 14$).

	Max	Min	Mean	Median
Number of Sessions	20	5	12.21	12.00
Active Usage Hours	32.1	2.7	11.50	8.61
Unique Command Count*	91	43	69.4	66
Average Popularity of Commands*	65.2%	52.0%	59.2%	60.1%
Command Completion Time (s)	7.838	1.906	4.237	3.701
Number of Undo Per Command	0.221	0.014	0.071	0.071
Number of Erase Per Command	0.232	0.001	0.115	0.122
Manual Commands Per Minute	6.896	1.932	3.308	2.941
Pauses Per Command	11.33	2.055	6.340	6.765
Menu Access Time (KLM Ratio)*	0.891	0.461	0.639	0.618
Percentage of Commands Accessed Through Menu	96.9%	10.9%	47.6%	41.3%

Note. **Rows marked with an asterisk were calculated from the 10 participants who had more than 6 hr of usage data.**

FIGURE 10. a) Correlation between unique commands used and overall expert assessment (N = 10). b) Correlation between in situ command completion time, and overall expert assessment.



Popularity of command is calculated as the percentage of users, from our log of about 3,000 users who use that command. However, this too did not have a correlation with expert assessments. This reiterates the data from the familiarity survey—*the number and nature of commands a user uses does not necessarily correlate to assessed expertise levels.*

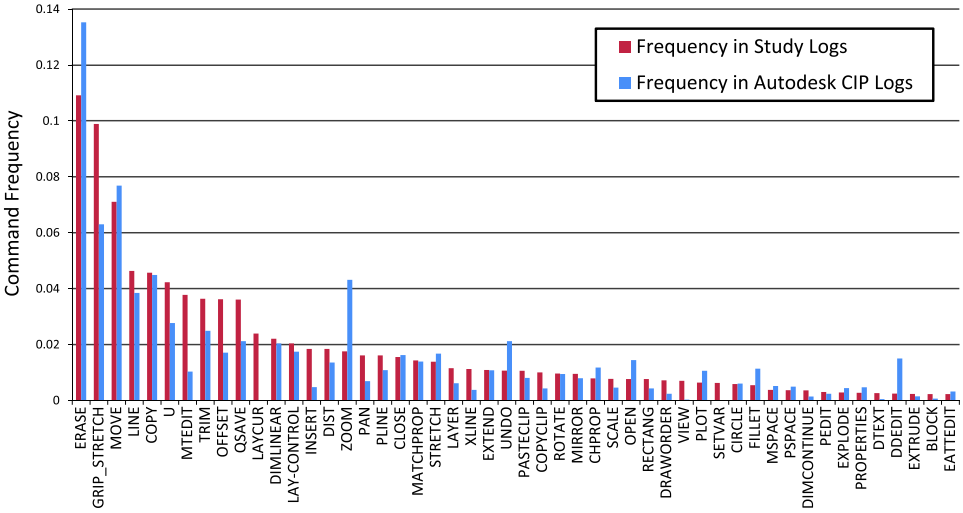
For example, P10 and P14 had lower command counts compared to other users with similar overall expert assessments (Figure 10a, highlighted). It is interesting to note these user's had similar backgrounds—about 6 years of experience, and an undergraduate degree in architecture. These users could be classified as *isolated experts*. It is also interesting to note that P3, who was assessed as a novice, had the highest number of unique commands used. This user only had 1 year of experience but recently obtained an advanced AutoCAD training certificate. So this user could be classified as a *knowledgeable expert*. We can also use the command frequencies to compare our participants' usage habits to the typical usage patterns seen in the actual AutoCAD user community.

Figure 11 illustrates the command frequencies across our participants, for the 50 most frequently used commands captured by our usage logs. These frequencies are compared to the actual command frequencies, as reported by AutoCAD's CIP program. It can be seen that the distribution is quite similar. The correlation is significant, with a near 0 intercept and near unity slope ($p < .0001$, $y = 0.9507x + 0.0008$, $r^2 = .8423$). This provides evidence that our study participants were a reasonable representation of actual AutoCAD users and that the 2-week collection period was sufficient in gathering representative usage patterns.

Command Efficiency

We next look at how efficient users are at executing commands. This is done by examining the time between when a command begins and finishes. In AutoCAD, commands have explicit start and end times—commands are initiated when the associated command is accessed from the user interface and completed once the command has been issued, typically by pressing enter.

FIGURE 11. A comparison of the command usage frequencies between our study participants and actual Autodesk CIP logs, for the 50 most frequently used commands in the study logs.



To correct the skewing and remove outliers, we calculated the median value for each user. There was significant correlation with command execution time and overall expert assessment ($r^2 = .49$, $p < .01$; Figure 10b). This was somewhat surprising, given the high level of noise we expected in the command usage times in situ, without any knowledge of the tasks or even the commands being used. *This is new evidence that command efficiency could be measured in situ and used as an indicator of expertise.*

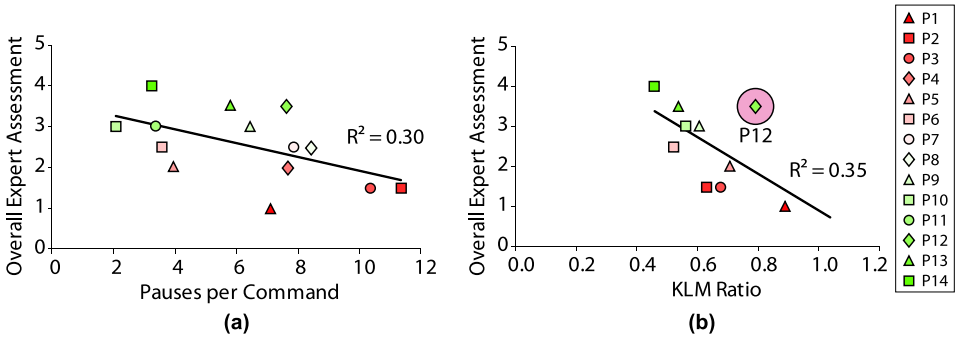
Undo and Erase

In addition to looking at these overall command metrics, we also wanted to look at two specific commands, undo and erase, that have been previously identified as commands that may be indicative of usability issues (Akers et al., 2009). Hence such commands may also be indicative of expertise. For each user, we measured the frequency of use of these two commands relative to the total number of commands that the user executed. As shown in Figure 9, the frequencies ranged greatly across users, from 1.4% to 22.1% for undo and 0.1% to 23.2% for erase. However the frequency of use with these commands did not have a significant correlation to users' assessed expertise levels ($p > .5$ in both cases). This indicates that novices and experts integrate these commands into their workflows in similar manners, and as such may not be indicative of expertise level.

Workflow Pace

We also looked at metrics corresponding to the user's pace: commands entered per minute, and pause frequency. Pauses were defined as any break in mouse or keyboard input that lasted longer than 300 ms (Ghazarian & Noorhosseini, 2010).

FIGURE 12. a) Correlation between pauses and overall expert assessment ($N = 14$). b) Correlation between menu access times and overall expert assessment ($N = 10$).



Such pauses could occur at any time while the AutoCAD was in focus. Pause frequency was calculated by dividing the number of pauses by the number of commands a user issued. Pause frequency had a significant correlation to overall expert assessment ($r^2 = .30$, $p < .05$; Figure 12a). Again, this is an important result, as we have validated another metric that may be used to identify expertise levels in situ, without any prior knowledge about the user or their tasks. From the GOMS literature, this could be explained by users requiring less time to perform mental operations, which is argued to be a sign of expertise (John & Kieras, 1996).

Commands entered per minute did not have a significant correlation with the expert assessments. Unlike pauses, previously discussed, this metric is highly dependent on the nature of the task, which may be why it is less indicative of expertise.

These results shows that the high level pace of a user may not be indicative of expertise, whereas low level pauses may indicate times where users need to consider their next steps, something which experts do less.

Menu Access

Guided by previous work (Hurst et al., 2007), we looked at menu access times in the usage logs. The metric we looked at is the one proposed previously, the Keystroke Level Model (KLM) ratio, where the actual menu acquisition time is divided by the expected acquisition time, as predicted by a KLM GOMS model (Hurst et al., 2007). We did not include the four participants with less than 6 hours of usage data, as these users all had less than 35 menu invocations. The initial analysis included menu items at all depths, but no significant correlations were found. We repeated the analysis with only top-level menu acquisitions, and there was a weak correlation between the KLM and the overall expert assessment ($r^2 = .35$, $p = .072$). Although this shows that there is some information within this metric, it is not completely reliable. For example, Figure 12b illustrates that P12 has a higher KLM ratio for his level of expertise. Upon further inspection, P12 has used the menu for the least number of commands, across all users. *As such, metrics which use menu access need to also consider how the user typically accesses commands.*

Access Metrics

In relation to this, we looked at the frequency at which users accessed the menu, command line, and shortcuts. Although we hypothesized that the expert users would rely more on accelerators (command line and short cuts), this was not the case. None of the aforementioned correlated to the expert assessments. Our observations in the laboratory study revealed that users have their own unique ways for accessing commands, regardless of their level of expertise. For example, P13, a user who was assessed at a high level of expertise, accessed only 21.5% of commands from the command line even though the command line is considered to be an efficient mechanism to access commands in AutoCAD.

5.3. Summary

We have presented a detailed analysis of the results from each phase of the study we conducted. We were able to identify a number of interesting patterns, and a number of metrics that correlated to the expert assessments. A full table of the correlation analysis is presented in Figure 13. The most notable results that we have found are as follows:

- Both years of experience and self-assessment of expertise levels may be misleading. Neither of these metrics correlated with the overall expert assessment levels.
- Users demonstrated a spectrum of usage frequencies with commands. The usage frequencies from the usage logs identified two *isolated experts* that used a small set of commands, but had a high-assessed level of expertise.
- The command distributions captured by our software logs closely matched frequencies from actual Autodesk logs from a large population of users. This demonstrates that our methodology of capturing data in situ is effective at obtaining representative samples of data.
- Although shown to be a strong metric previously (Hurst et al., 2007), menu access times in our laboratory study only had a weak correlation with the overall expert assessment ($p = .072$). In particular, there were users who typically used other command access techniques and had slower times with the menu.
- Performance with individual commands, as measured in our laboratory study, exhibited the strongest correlation, across all metrics, with the overall expert assessment. In addition, the data from completion times indicated the presence of *naïve experts* (fast with the commands but lower expertise ratings). Surprisingly, in situ command completion times also correlated to the overall expert assessment—providing the new evidence that software expertise could potentially be measured in situ, without any prior knowledge.
- A higher number of pauses in situ correlated to the overall expert assessment. However, command usage rates did not, indicating that to be an expert does not necessarily mean to work faster.

FIGURE 13. Summary of the correlation analysis from the study results.

Measure	Collection Method (Phase #)	2D Expert Assessment	3D Expert Assessment	Overall Expert Assessment
Unique Command Count	In-Situ Metrics Phase 1	$R^2 = 0.13$ ($p = 0.1958$)	$R^2 = 0.01$ ($p = 0.6997$)	$R^2 = 0.13$ ($p = 0.1924$)
Average Popularity of Commands		$R^2 = 3.47$ ($p = 0.9840$)	$R^2 = 0.01$ ($p = 0.6563$)	$R^2 = 0.00$ ($p = 0.9558$)
Command Completion Time		**$R^2 = 0.39$ ($p = 0.0169$)****	$R^2 = 0.19$ ($p = 0.1192$)	**$R^2 = 0.49$ ($p = 0.0053$)****
Number of Undo Per Command		$R^2 = 0.15$ ($p = 0.1690$)	$R^2 = 0.05$ ($p = 0.4148$)	$R^2 = 0.02$ ($p = 0.6032$)
Number of Erase Per Command		$R^2 = 0.01$ ($p = 0.7972$)	$R^2 = 0.01$ ($p = 0.8633$)	$R^2 = 0.01$ ($p = 0.8032$)
Pauses Per Command		**$R^2 = 0.45$ ($p = 0.0084$)****	$R^2 = 0.12$ ($p = 0.2076$)	**$R^2 = 0.29$ ($p = 0.0444$)****
Manual Commands Per Minute		$R^2 = 0.15$ ($p = 0.1705$)	$R^2 = 0.01$ ($p = 0.7580$)	$R^2 = 0.04$ ($p = 0.4776$)
Menu Access Time (KLM Ratio)		$R^2 = 0.22$ ($p = 0.1714$)	$R^2 = 0.10$ ($p = 0.3733$)	$R^2 = 0.35$ ($p = 0.0716$)*
Command Line Frequency		$R^2 = 0.02$ ($p = 0.5975$)	$R^2 = 0.01$ ($p = 0.7193$)	$R^2 = 0.01$ ($p = 0.9174$)
Shortcut Frequency		$R^2 = 0.12$ ($p = 0.2089$)	$R^2 = 0.01$ ($p = 0.9532$)	$R^2 = 0.08$ ($p = 0.3068$)
Experience (Years)	Self-Assessment Phase 2	$R^2 = 0.07$ ($p = 0.3422$)	$R^2 = 0.02$ ($p = 0.6243$)	$R^2 = 0.24$ ($p = 0.0695$)*
Self-Assessed Expertise		$R^2 = 0.04$ ($p = 0.4622$)	$R^2 = 0.01$ ($p = 0.7427$)	$R^2 = 0.03$ ($p = 0.5186$)
Commands Used	Self-Assessment Phase 3	$R^2 = 0.06$ ($p = 0.3875$)	$R^2 = 0.09$ ($p = 0.2959$)	$R^2 = 0.09$ ($p = 0.2786$)
Commands Familiar With		$R^2 = 0.01$ ($p = 0.8778$)	$R^2 = 0.01$ ($p = 0.7325$)	$R^2 = 0.01$ ($p = 0.8348$)
3D Commands Used		$R^2 = 0.10$ ($p = 0.2605$)	**$R^2 = 0.31$ ($p = 0.0361$)****	$R^2 = 0.19$ ($p = 0.1125$)
UI Access Time	UI Access Study Phase 4	**$R^2 = 0.53$ ($p = 0.0031$)****	$R^2 = 0.16$ ($p = 0.1495$)	$R^2 = 0.26$ ($p = 0.0624$)*
Tool Task Completion Score	Tool Task Study Phase 5	**$R^2 = 0.58$ ($p = 0.0014$)****	**$R^2 = 0.34$ ($p = 0.0283$)****	**$R^2 = 0.79$ ($p < 0.0001$)****
Tool Task Completion Time		**$R^2 = 0.39$ ($p = 0.0164$)****	$R^2 = 0.11$ ($p = 0.2386$)	$R^2 = 0.23$ ($p = 0.0778$)*

** $p < 0.05$ | * $p < 0.10$

6. DISCUSSION

We have performed a thorough investigation of metrics for application-level software expertise. This included presenting a framework within which expertise metrics can be defined. In addition, we performed a study that examined the various dimensions of the framework, looked at how the metrics correlate with one another, and examined how well metrics of usage, including those gathered in situ, correlate to expertise.

We believe the framework we presented will be a useful reference going forward. Our survey of previous research indicated a number of proposals as to what a software expert was, including a number of definitions which only looked at lower level components. We made an explicit effort to capture this diversity in our framework of possible metrics. This framework is not meant to define expertise, it is meant to help ground discussions on expertise and provide a space for which metrics could be chosen. In addition to contributing a new framework of application expertise metrics, we see our study as providing two contributions to the existing HCI literature:

- It serves as the first validation that previously proposed metrics not only apply to low-level and short-term expertise but also high-level software expertise.
- It serves as the first validation that metrics from in situ usage data, without any knowledge about the user environment or task, can be correlated to software expertise.

6.1. Design Implications

Typically expertise detection is motivated by the use of adaptive user interfaces (Carroll & Carrithers, 1984; Shneiderman, 2003) or help systems (Heckerman & Horvitz, 1998; Hurst et al., 2007). However, by detecting expertise at the application level, we see some additional opportunities for design. First, we believe it will be valuable for community-based learning resources, such as online discussion boards, for users to see and understand the nature of other users' expertise levels. Similarly, techniques that make software command recommendations on other users' usage behaviors (Linton & Schaefer, 2000; Matejka et al., 2009) could give higher weight to the relevant experts.

We also see organizational implications for understanding software expertise at the application level. If a software company could use its own software to understand the spectrum of expertise of its end-users, it could better understand what types of learning aids it should focus on developing. By dynamically measuring progress in users' expertise at the application level, a company could additionally track the impact of a new learning aid or software revision.

Similarly, a company that has a large number of employees using the same piece of software, such as an architecture firm or graphic design studio, could track the expertise levels of its employees. This could allow the company to make informed decisions about purchasing training materials or asking specific employees to attend specialized training courses.

6.2. Limitations and Future Work

Overall, our study was both extensive and exploratory. The study consisted of six different phases, each with its own protocol, and it required student and professional users to install a plug-in, on their home or office workstations. Although we were able to identify certain significant correlations, our results do still warrant future investigation and validation. In particular several challenges in executing this work led to both limitations and opportunities for future work, which we discuss next.

Generality of Results

Overall, there are a number of cautions that must be taken into account before generalizing our results. The main issues that should be considered are the number of participants used, the target application, and the limited number of tasks that were used during the expert assessments. We discuss each of these issues in greater detail next.

Number of Participants

Our analysis was based on 14 participants who completed the study. This served our purpose of providing an initial investigation of software expertise, and a larger scale study may be able to better identify different expertise profiles and provide stronger validations of some of our findings. However, even with the number of participants we worked with, we were still able to identify some interesting trends and unique characteristics from our data set.

Generalization to Other Software

Our framework was developed based on a literature review of expertise across many different domains of software and should generalize to almost any end-user software application. However, our study was conducted with a single target application, AutoCAD. Some aspects of AutoCAD are unique, which may impact the generalization of our study analysis. For example, users explicitly start and end commands, which allowed us to accurately capture command completion times. Although some of these application specific features were used in our analysis, other metrics were based on more general software features. As such, we feel our results do generalize, but validation against other software applications would be an interesting project for future research.

Expert Assessment Tasks

Another caution that should be considered is that our expert judges made their judgments based on observing only four predetermined tasks. Although this may seem like a small number of tasks, it did provide for approximately 1 hr of usage observations by the expert judges. Furthermore, these four tasks were specifically developed by our expert judges, who both have extensive experience assessing skill levels of AutoCAD users. They intentionally developed the tasks in a way that would allow them to make their assessments. The tasks were high level enough that there were many ways a participant could go about accomplishing the task. As such, the experts were not just assessing how well the participants used a sequence commands to accomplish the tasks but the overall approach and strategies that the participants used.

At the end of our study we discussed this issue with our expert judges to gauge their sense of confidence in their ratings. Both responded that they had high confidence in their assessments, indicating that the four tasks, across a 1-hr period, were adequate for forming their assessments. However, an interesting future topic would be to investigate how an expert's assessment of a user's expertise level evolves, depending on the duration of the observation period.

Correlation Analysis

Our analysis was performed by collecting multiple measures of expertise and investigating the relationship among those measures. This methodology is similar to analyses of convergent validity used in psychometric research (Messick, 1995),

where one is trying to understand what a particular metric is measuring by looking at correlations among a variety of other metrics. Although a number of the correlations were found to be significant, it only indicates that *part* of the variance in expertise can be explained by the metric, with other factors at play. In addition, we tested multiple correlations without pairwise correction. For studies such as ours, which are of an exploratory nature, this is an accepted statistical protocol (Curtin & Schulz, 1998), but it is important to recognize that this does increase chances of Type I errors, so our results should be generalized and contemplated with caution.

True Novices

Because our methodology included in situ usage data, we required participants to be existing AutoCAD users. As such, our study did not focus on the detection of true novices using the application for the first time. However, our framework of metrics should also be applicable to novice users, and it would be interesting to consider how this framework could be used to provide classifications of types of novice users, similar to the different classifications of expertise outlined in Section 4.1.

Automatic Classification

We have provided useful information as to what features may be used to identify a user's expertise levels. This was an initial study, and a next step could be to create statistical expertise classifiers (Hurst et al., 2007) based on application usage logs. A larger user base will be needed before such a technique can be validated, but our work sets important groundwork for such an undertaking.

Domain Knowledge

With our study, we tried to investigate as much of our framework of software expertise as possible. However, one omission was the issue of domain knowledge. Although we collected some information about education experience, it would be hard to accurately capture each of the participants' knowledge level of architecture and design at a broad level. Although identifying a user's domain knowledge would be challenging, we still feel it is an important component of the framework, as it is likely to influence how a user works with a software system (Kolodner, 1983; Nielsen, 1994). In addition to domain knowledge, it may also be interesting in the future to consider knowledge about computers, software, and operating systems in general, in relation to the target application. In particular, a user with stronger knowledge of such issues may be more likely to achieve expertise levels within certain areas of a software application, such as importing or exporting data to and from other applications.

NOTES

Acknowledgments. We thank Ramtin Attar and Mehrdad Tavakkolian for their assistance with the laboratory study and their guidance on the task designs and John Yee for

assisting with the study setup. We also thank the study participants. Finally, we thank the reviewers of this article. In particular, we acknowledge Dick Pew's efforts as the Action Editor for our article, whose comments and suggestions helped us significantly improve the final quality of our manuscript.

HCI Editorial Record. First received on March 8, 2013. Revisions received on July 5, 2013, October 29, 2013, and November 29, 2013. Accepted by Dick Pew. Final manuscript received on December 19, 2013. — *Editor*

REFERENCES

Adar, E., Teevan, J., & Dumais, S. T. (2008). Large-scale analysis of web revisitation patterns. *Proceedings of the CHI 2008 Conference on Human Factors in Computer Systems*. New York: ACM.

Akers, D., Simpson, M., Jeffries, R., & Winograd, T. (2009). Undo and erase events as indicators of usability problems. *Proceedings of the CHI 2009 Conference on Human Factors in Computer Systems*. New York: ACM.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*, 167–207.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Erlbaum.

Bhavnani, S., & Bates, M. (2002). Separating the knowledge layers: Cognitive analysis of search knowledge through hierarchical goal decompositions. *Proceedings of the 2002 ASIST Annual Meeting*. New York, NY: Wiley.

Bhavnani, S. K., & John, B. E. (1996). Exploring the unrealized potential of computer-aided drafting. *Proceedings of the CHI 1996 Conference on Human Factors in Computer Systems*. New York: ACM.

Bhavnani, S. K., & John, B. E. (1997). From sufficient to efficient usage: An analysis of strategic knowledge. *Proceedings of the CHI 1997 Conference on Human Factors in Computer Systems*. New York: ACM.

Bhavnani, S. K., & John, B. E. (1998). Delegation and circumvention: Two faces of efficiency. *Proceedings of the CHI 1998 Conference on Human Factors in Computer Systems*. New York: ACM.

Bhavnani, S. K., & John, B. E. (2000). The strategic use of complex computer systems. *Human Computer Interaction, 15*, 107–137.

Bhavnani, S. K., John, B. E., & Flemming, U. (1999). The strategic use of CAD: An empirically inspired, theory-based course. *Proceedings of the CHI 1999 Conference on Human Factors in Computer Systems*. New York: ACM.

Card, S. K., English, W. K., & Burr, B. J. (1987). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT. In *Human–computer interaction: A multidisciplinary approach* (pp. 386–392). Burlington, MA: Morgan Kaufmann.

Card, S. K., Moran, T., & Newell, A. (1983). *The psychology of human–computer interaction*. Hillsdale, NJ: Erlbaum.

Carroll, J. M., & Carrithers, C. (1984). Training wheels in a user interface. *Communications of the ACM, 27*, 800–806.

Chapuis, O., Blanch, R., & Beaudouin-Lafon, M. (2007). *Fitts' law in the wild: A field study of aimed movements* (Tech. Rep. No. 1480). Paris: LRI, Univ. Paris-Sud, France.

Collins, H. M., & Evans, R. (2002). The third wave of science studies: Studies of expertise and experience. *Social Studies of Science, 32*, 235–296.

Cote-Munoz, J. (1993). AIDA—An adaptive system for interactive drafting and CAD applications. *Human Factors in Information Technology, 10*, 225–225.

Curtin, F., & Schulz, P. (1998). Multiple correlations and Bonferonni's correction. *Biological Psychiatry, 3*, 775–777.

Dreyfus, H. L., Anthanasiou, T., & Dreyfus, S. E. (2000). *Mind over machine: The power of human intuition and expertise in the era of the computer*. New York, NY: Simon & Schuster.

Einhorn, H. J. (1974). Expert judgment: Some necessary conditions and an example. *Journal of Applied Psychology, 59*, 562–571.

Ericsson, K. A., Krampe, R. T.., & Tesch-Romer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100*, 363–406.

Ericsson, K. A., & Lehmann, A. C. (1996). Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual Review of Psychology, 47*, 273–305.

Ericsson, K. A. (2006). *The Cambridge handbook of expertise and expert performance*. Cambridge, UK: Cambridge University Press.

Ericsson, K. A., & Williams, A. M. (2007). Capturing naturally occurring superior performance in the laboratory: Translational research on expert performance. *Journal of Experimental Psychology: Applied, 13*, 115–123.

Evans, A., & Wobbrock, J. (2012). Taming wild behavior: The input observer for obtaining text entry and mouse pointing measures from everyday computer use. *Proceedings of the CHI 2012 Conference on Human Factors in Computer Systems*. New York: ACM.

Fernquist, J., Grossman, T., & Fitzmaurice, G. (2011). Sketch–sketch revolution: An engaging tutorial system for guided sketching and application learning. *Proceedings of the UIST 2011 Symposium on User Interface Software and Technology*. New York: ACM.

Findlater, L., & McGrenere, J. (2007). Evaluating reduced-functionality interfaces according to feature findability and awareness. *Proceedings of the IFIP INTERACT 2007*. Berlin, Germany: Springer.

Findlater, L., & McGrenere, J. (2010). Beyond performance: Feature awareness in personalized interfaces. *International Journal of Human–Computer Studies, 68*, 121–137.

Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., & Weld, D. S. (2008). Predictability and accuracy in adaptive user interfaces. *Proceedings of the CHI 2008 Conference on Human Factors in Computer Systems*. New York: ACM.

Gajos, K., Reinecke, K., & Herrmann, C. (2012). Accurate measurements of pointing performance from in situ observations. *Proceedings of the CHI 2012 Conference on Human Factors in Computer Systems*. New York: ACM.

Ghazarian, A., & Noorhosseini, S. M. (2010). Automatic detection of users' skill levels using high-frequency user interface events. *User Modeling and User-Adapted Interaction, 20*, 109–146.

Gray, W. D., John, B. E., & Atwood, M. E. (1992). The precis of Project Ernestine or an overview of a validation of GOMS. *Proceedings of the CHI 1992 Conference on Human Factors in Computer Systems*. New York: ACM.

Greenberg, S., & Witten, I. H. (1988). Directing the user interface: how people use command-based computer systems. *Proceedings of the IFAC 1988 Conference on Man-Machine Systems*. Laxenburg, Austria: International Federation of Automatic Control.

Grossman, T., Dragicevic, P., & Balakrishnan, R. (2007). Strategies for accelerating on-line learning of hotkeys. *Proceedings of the CHI 2007 Conference on Human Factors in Computer Systems*. New York: ACM.

Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: Metrics, methodologies and guidelines. *Proceedings of the CHI 2009 Conference on Human Factors in Computer Systems*. New York: ACM.

Heckerman, D., & Horvitz, E. (1998). The Lumiere Project: Bayesian user modeling for inferring the goals and needs of software users. *Proceedings of the 1998 Conference on Uncertainty in Artificial Intelligence*. Burlington, MA: Morgan Kaufmann.

Hurst, A., Hudson, S. E., & Mankoff, J. (2007). Dynamic detection of novice vs. skilled use without a task model. *Proceedings of the CHI 2007 Conference on Human Factors in Computer Systems*. New York: ACM.

Hurst, A., Mankoff, J., & Hudson, S. E. (2008). Understanding pointing problems in real world computing environments. *Proceedings of the 2008 SIGACCESS Conference on Computers and Accessibility*. New York: ACM.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer–Human Interaction, 3*, 320–351.

Kolodner, J. L. (1983). Towards an understanding of the role of experience in the evolution from novice to expert. *International Journal of Man–Machine Studies, 19*, 497–518.

Kurtenbach, G. (1993). *The design and evaluation of marking menus*. Unpublished doctoral dissertation, University of Toronto, Toronto, Canada.

LaFrance, M. (1989). The quality of expertise: Implications of expert–novice differences for knowledge acquisition. *ACM SIGART Bulletin, 108*, 6–14.

Lafreniere, B., Bunt, A., Whissell, J. S., Clarke, C. L. A., & Terry, M. (2010). Characterizing large-scale use of a direct manipulation application in the wild. *Proceedings of Graphics Interface*. Canadian Information Processing Society.

Lang, G. T., Eberts, R., Gabel, M., & Barash, M. (1991). Extracting and using procedural knowledge in a CAD task. *IEEE Transactions on Engineering Management, 38*, 257–268.

Lawson, B. R., Baker, K. R., Powell, S. G., & Foster-Johnson, L. (2009). A comparison of spreadsheet users with different levels of experience. *Omega, 37*, 579–590.

Lee, F. J., & Anderson, J. R. (2001). Does learning of a complex task have to be complex? A study in learning decomposition. *Cognitive Psychology, 42*, 267–316.

Li, W., Matejka, J., Grossman, T., Konstan, J. A., & Fitzmaurice, G. (2011). Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer–Human Interaction, 18*(2), 1–35.

Linton, F., Joy, D., Schaefer, H. P., & Charron, A. (2000). OWL: A recommender system for organization-wide learning. *Educational Technology & Society, 3*, 62–76.

Linton, F., & Schaefer, H.-P. (2000). Recommender systems for learning: Building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction, 10*, 181–208.

Masarakal, M. (2010). *Improving expertise-sensitive help systems*. Unpublished master's thesis, University of Saskatchewan, Saskatoon, Canada.

Matejka, J., W. Li, T. Grossman & G. Fitzmaurice (2009). CommunityCommands: Command recommendations for software applications. *Proceedings of the UIST 2009 Symposium on User Interface Software and Technology*. New York: ACM.

McGrenere, J., & Moore, G. (2000). Are we all in the same "bloat"? *Proceedings of Graphics Interface*. Calgary, Alberta, Canada: Canadian Information Processing Society.

Messick, S. (1995). Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientific inquiry into score meaning. *American Psychologist, 50*, 741–749.

Nielsen, J. (1994). *Usability engineering*. Burlington, MA: Morgan Kaufmann.

Nilsen, E., Jong, H., Olson, J. S., Biolsi, K., Rueter, H., & Mutter, S. (1993). The growth of software skill: A longitudinal look at learning & performance. *Proceedings of the CHI 1993 Conference on Human Factors in Computer Systems*. New York: ACM.

Nisbett, R. E., & Wilson, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review, 84*, 231–259.

Norman, K. L. (1991). *The psychology of menu selection: Designing cognitive control at the human/computer interface*. Westport, CT: Greenwood Press.

Paris, C. L. (1988). Tailoring object descriptions to a user's level of expertise. *Computational Linguistics, 14*(3), 64–78.

Pemberton, J. D., & Robson, A. J. (2000). Spreadsheets in business. *Industrial Management & Data Systems, 100*, 379–388.

Rasmussen, J. (1987). Skills, rules, and knowledge: Signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics, 3*, 257–266.

Rooke, M., Grossman, T., & Fitzmaurice, G. (2011). AppMap: Exploring user interface visualizations. *Proceedings of Graphics Interface*. Calgary, Alberta, Canada: Canadian Information Processing Society.

Scarr, J., Cockburn, A., Gutwin, C., & Bunt, A. (2012). Improving command selection with CommandMaps. *Proceeding of CHI 2012 Conference on Human Factors in Computer Systems*. New York: ACM.

Scarr, J., Cockburn, A., Gutwin, C., & Quinn, P. (2011). Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. *Proceedings of the CHI 2011 Conference on Human Factors in Computer Systems*. New York: ACM.

Shaughnessy, H. (2013, June 17). Autodesk—120 million reasons why the future lies with makers. *Forbes*. Retrieved from http://www.forbes.com/

Shneiderman, B. (2003). Promoting universal usability with multi-layer interface design. *Proceedings of the SIGCAPH 2003 Conference on Universal Usability*. New York: ACM.

Teevan, J., Dumais, S. T., & Horvitz, E. (2005). Personalizing search via automated analysis of interests and activities. *Proceedings of the SIGIR 2005 Conference on Research and Development in Information Retrieval*. New York, NY: ACM.

Teevan, J., Dumais, S. T., & Liebling, D. J. (2008). To personalize or not to personalize: Modeling queries with variation in user intent. *Proceedings of the SIGIR 2008 Conference on Research and Development in Information Retrieval*. New York: ACM.

Teevan, J., Dumais, S. T., & Liebling, D. J. (2010). A longitudinal study of how highlighting web content change affects people's web interactions. *Proceeding of CHI 2010 Conference on Human Factors in Computer Systems*. New York: ACM.

Thorndike, E. L. (1921). *The psychology of learning* (Vol. 2). New York, NY: Teachers College, Columbia University.

VanLehn, K. (1989). Problem solving and cognitive skill acquisition. In M. I. Posner (Ed.), *Foundations of cognitive science* (pp. 527–579). Cambridge, MA: MIT Press.

VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology, 47*, 513–539.

White, R. W., Dumais, S. T., & Teevan, J. (2009). Characterizing the influence of domain expertise on web search behavior. *Proceedings of the 2009 International Conference on Web Search and Data Mining*. New York: ACM.