Characterizing Large-Scale Use of a Direct Manipulation Application in the Wild

Benjamin Lafreniere [1]
HCI Lab
University of Waterloo

Andrea Bunt [2]
HCI Lab
University of Manitoba

John S. Whissell [3]
AI Research Group
University of Waterloo

Charles L. A. Clarke [4]
Information Retrieval Group
University of Waterloo

Michael Terry [5]
HCI Lab
University of Waterloo

ABSTRACT

Examining large-scale, long-term application use is critical to understanding how an application meets the needs of its user community. However, there have been few published analyses of long-term use of desktop applications, and none that have examined applications that support creating and modifying content using direct manipulation. In this paper, we present an analysis of 2 years of usage data from an instrumented version of the GNU Image Manipulation Program, including data from over 200 users. In the course of our analysis, we show that previous findings concerning the sparseness of command use and idiosyncrasy of users' command vocabularies extend to a new domain and interaction style. These findings motivate continued research in adaptive and mixed-initiative interfaces. We also describe the novel application of a clustering technique to characterize a user community's higher-level tasks from low-level logging data.

KEYWORDS: Logging, long-term usage, community command usage, open source software, remote usability, adaptive interfaces, longitudinal study

INDEX TERMS: H.5.m [Information Interfaces and Presentation (e.g., HCI)]: Miscellaneous

1 INTRODUCTION

Modern content-creation applications contain hundreds of features and are used by millions of users for a multitude of tasks. However, there is little published work characterizing large-scale, long-term usage patterns of these types of applications. Without this information, it is difficult to know how these feature-rich user interfaces should evolve to meet the needs of users.

In this paper, we present an analysis of the ingimp dataset, a dataset collected from a public deployment of ingimp, an instrumented version of the GNU Image Manipulation Program (GIMP) [18]. The dataset represents more than 200 users and 4000 sessions collected over a 2-year period. This paper's contributions include the analysis of this dataset, the novel methods employed to perform this analysis, and a discussion of implications for interface design.

Our analysis of this dataset represents the first large-scale analysis of a content-creation application that deals primarily with non-text content (i.e., graphics). It is also the first such analysis of an application for which direct manipulation is the primary mode of interaction. In contrast, past analyses have focused exclusively on applications that largely deal with text and keyboard input, such as word processors (e.g., [3,10,12,21]) or software development environments [14]. This work complements and extends previous research by describing the common patterns and trends of long-term use in an alternative application space.

Despite having a qualitatively different style of interaction, we find that many previous findings for text-centric applications apply to ingimp as well. For example, we find that users' command vocabularies tend to be small (averaging only 27 commands of the nearly 500 available) and have little overlap with one another (the vast majority of commands are used by no more than 10% of users). We also find that users tend to use only a small portion of the available functionality (only about 6 unique commands in any particular session). These results have implications for interface design strategies intended to mitigate application complexity, such as adaptive or mixed-initiative interfaces, and motivate the need for further research in these spaces.

Our data analysis also uncovered limitations in the metrics commonly used to summarize large-scale application use by a user community. Past analyses have characterized application use through frequency counts of commands across the entire user base (e.g. [6,10]). Direct manipulation tools, however, skew this metric because tools such as the paintbrush may be used dozens or hundreds of times in succession, with each stroke logged as an individual command invocation. This work presents a set of additional perspectives to derive a richer picture of command use across a community of users. Specifically, we consider the number of users who have used a command, the number of sessions (log files) in which a command has appeared, and command use with repeated invocations collapsed to a single invocation. Respectively, these perspectives indicate a command's relative popularity in the community, its relative importance across *all* tasks, and its importance within a particular session, thus providing a more holistic picture of application usage than raw command counts alone.

Finally, previous long-term studies of content-creation applications have tended to characterize application use at a relatively fine level of granularity, often going no deeper than the aforementioned command counts across the entire user community. While such analyses provide important insights into large-scale application use, they do not easily translate to an understanding of the higher-level tasks performed by users. Toward this end, we demonstrate the novel application of a clustering technique to automatically derive approximations of common tasks performed by the user community. This strategy yields sets of commands that clearly relate to particular tasks, such as photo retouching or graphic design work. Moreover, we show that the resultant sets of commands correspond well with user-supplied descriptions of their intended tasks, validating the overall approach.

The rest of this paper is structured as follows. We start by reviewing related work on long-term application usage. Next, we describe ingimp and the ingimp dataset. We provide a basic summary of ingimp users, including the characteristics of their sessions (e.g., length, frequency of use) and the characteristics of the documents that they worked on. We then turn our attention to summarizing application use by considering the popularity of different commands by a number of metrics. Finally, we characterize user tasks through the previously mentioned clustering technique and show that user-supplied descriptions of

[1]bjlafren@cs.uwaterloo.ca, [2]bunt@cs.umanitoba.ca,
[3]jswhisse@cs.uwaterloo.ca, [4]claclarke@plg.uwaterloo.ca,
[5]mterry@cs.uwaterloo.ca

tasks validate this analytical approach. We conclude with a discussion of the implications of our findings.

2 Related Work

In this paper, we are primarily concerned with characterizing long-term use of *content-creation desktop applications*. While we recognize the extensive literature studying large-scale use of games (e.g., [5]) and web applications (e.g., [11,16]), in this research we are motivated to understand how feature-rich software is applied to ill-defined tasks. Past work in this particular space has examined long-term use of the UNIX command line, text editors, and software development environments, which we review below.

Greenberg analyzed data on the commands issued by 168 users in a UNIX command-line environment over a four-month period [6]. This research replicated a number of previous studies analyzing UNIX command use, including that of Draper [4] and Hanson *et al*. [7]. In his work, Greenberg found that individuals used only a very small number of the available commands (a mean of 50 of the 1307 unique commands observed during the monitoring period). Furthermore, he found very little overlap between individual users' *command vocabularies*—the set of unique commands a user was observed using—with less than 3% of commands shared by more than 50% of users. Finally, Greenberg found that individual users were likely to reuse commands that they had recently used, motivating the design of intelligent history mechanisms.

In the domain of text editing, Whiteside *et al*. [21] logged keystroke-level data for two text-editor applications, one used by six secretaries, the other by eight knowledge workers, over periods of 11 days and 2 months, respectively. The authors analyzed the frequency of individual keystrokes and transition probabilities between them. They found that 50% of users' keystrokes were used for text entry and 25% were used for cursor movement. The degree to which keystrokes were used for cursor movement led Digital Equipment Corporation (DEC) to adopt the now-familiar inverted-T layout for cursor keys on their VT200 series terminals. Studies such as these underscore the positive impact that can result from understanding long-term, real-world use of applications.

Studies of long-term application use have also informed the design of intelligent teaching systems. In a study that spanned three years, researchers instrumented the sam text editor to gather usage data from 2200 undergraduate Computer Science students at the University of Sydney, Australia [3,10]. By examining the logs of 63 students who used the application for the full three years, the researchers were able to build models of individual users' knowledge suitable to support teaching systems.

Also motivated by the goal of building models of user expertise, Linton *et al*. analyzed usage logs from 16 users of Microsoft Word 6.0 over periods ranging from 3 to 11 months per user (mean 6 months) [12]. While they found that users' command vocabularies are relatively small (a mean of 152 of the available 642 commands), they also found that the size of a users' command vocabulary is a poor measure of expertise since it is highly correlated with the length of time that the user has been observed.

Finally, Murphy *et al*. analyzed usage logs for 41 developers using Eclipse 3.1 and 3.2, with a focus on use of Java Development Tools (JDT) [14]. Usage logs were gathered for periods ranging from six to 125 days. This work provided the first account of how complex IDEs are used in the field, characterizing the frequency of use for various features and commands, and the work habits of Eclipse users.

One of the most common trends found in past work is that command frequencies, when considered across an entire community of users, follow a long-tailed distribution in which a very small number of commands account for the vast majority of observed command use, with the remaining commands used very little [3,4,6,7,10,12,21]. At the same time, individual command vocabularies are found to be relatively small and idiosyncratic, with little overlap between users. As we will show, the ingimp dataset follows both of these trends closely.

Finally, we note that past work has primarily focused on modeling user interaction at a relatively low-level. When higher-level tasks have been considered, researchers have often manually built models and examined how users conformed to them (see [9] for a discussion). We note an opportunity to apply clustering techniques to automatically extract likely tasks using command histories alone. Similar techniques have been successfully applied to classifying web logs according to user type (e.g, "German users"), information goal (e.g., "Support"), or task (e.g. "Online shopping") [8]. However, to our knowledge this type of higher-level task extraction has not been applied to desktop applications or to usage data from content-creation applications, where tasks are much less well-defined than those typical of web use.

Given this background, we now describe the ingimp dataset.

3 The ingimp Dataset

ingimp is an instrumented version of the open source GNU Image Manipulation Program [18]. All collected data is made publicly available on the project's website. ingimp was designed to collect the following information:

- Activity tags: optional user-supplied keywords describing how the user intends to use the application (prompted for at the start of each session)
- System characteristics, including operating system, CPU, number of monitors, monitor resolution, and time zone
- Document summaries, including the resolution and number of layers in the image
- Commands that appear on the undo stack

Users' locales are not explicitly recorded, but can be deduced by examining the localized command names in log files to determine the likely locale. While this method does a reasonable job of identifying users' locales, it does not perfectly differentiate between all locales (e.g., Canadian vs. British English).

Users' logs are automatically sent to the server when the application closes. We consider each such instance of the application being opened and closed a *session*. To permit tracking of users' activities across sessions, each user is assigned a randomly-generated ID number when ingimp is first run, which is subsequently included in the log of each session.

In the analyses that follow, we utilize the median and interquartile range (IQR) statistics when summarizing data, as the data do not follow normal distributions. However, we also include the mean and SD to facilitate comparison with prior work.

3.1 ingimp Deployment and Distribution

ingimp was announced in 2007 at an open source graphics conference, the Libre Graphics Meeting (LGM), and is freely available for download from www.ingimp.org. Since ingimp is released under the GNU Public License (GPL), anyone is free to download, install, and use it. Following its announcement, ingimp was featured on a number of websites, including Slashdot, GIMP's French language project page, and a story published on an open source-themed news site. Of the various announcements, the Slashdot coverage had the greatest single impact in user uptake, rapidly increasing the user base in a week's time.

Given the factors discussed above, ingimp's user base can be roughly approximated as Slashdot readers, GIMP users in general (with a slight emphasis on French users due to the French announcement), and those sympathetic to open source software. We

provide more specific details of the user base as we analyze the collected data.

In the sections that follow, we analyze the ingimp dataset as follows:

- We characterize the user community, including users' locales, time zones, computing environments, session lengths, and documents,
- We analyze command use by the community using a number of metrics, and
- We use an automatic cluster analysis technique to analyze the high-level tasks that users perform, and validate this analysis with user-supplied task descriptions

Our analysis considers all log files collected in the 2-year period between May 15, 2007 and May 15, 2009.

4 The ingimp User Base

We begin our analysis by considering characteristics of the user base. One complication in doing so, however, is the presence of *curious bystanders*. By definition, open source software can be downloaded, installed, and used by anyone. As a result, it can be expected that a number of people will try out the application out of curiosity alone. These users may not use the application after an initial test, making it worthwhile to filter them out before doing more in-depth analysis. We start by describing our criteria for filtering out these users, and then analyze the remaining users.

4.1 Defining Significant Users

For the ingimp dataset, we define a *significant user* as anyone who has used the application and saved a document on at least two separate days. Using these criteria, the ingimp community consists of 211 significant users who have contributed 4198 logs. This group constitutes only 22% of all ingimp users, though these users produce 75% of the log files (4198 out of 5612). While this may exclude some legitimate users, the fact that this minority accounts for 75% of all log files indicates that the criteria are reasonable. All subsequent data analyses use this significant user criterion to pre-filter the log files. Logs from the primary developers and researchers are also excluded from data analyses.

4.2 Users' Locales and Computing Environments

Time zone and locale information indicate that ingimp users are both geographically distributed and culturally diverse: A total of 15 different time zones and at least 9 different locales were recorded (we say "at least" because of the previously mentioned difficulties in precisely determining a user's locale in all situations). The vast majority of users use an English (59%) or French (31%) locale. The remaining users' locales include Italian, German, Spanish, Russian, Czech, Finnish, and Japanese.

The majority of ingimp users run Windows (73%) with the second-largest group running Linux (26%). Almost all participants (99%) use only one monitor, with the remaining 1% using 2 monitors. Screen resolutions vary greatly, ranging from 800x600 to 3840x1200, with a total of 37 different resolutions. The resolutions used by more than 10% of users are 1280x1024 (26%), 1024x768 (18%), and 1280x800 (10%).

4.3 Characterizing Users' Sessions

As previously mentioned, a session corresponds to activity logged between the time the application is opened and when it is closed. The median number of sessions for an individual user was 11 (IQR 15.5, mean 20, SD 26.0). As the large measures of spread indicate, the number of sessions for individual users varied widely, ranging from 1 to 168 (the significant user criteria were applied to the entire dataset, not just the dataset analyzed, leading to some single-session users in our analysis).

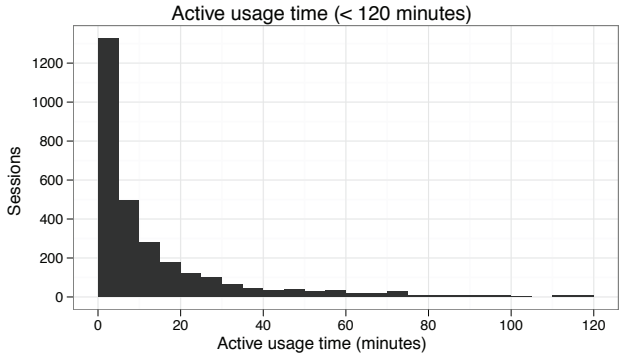

Figure 1. Histogram showing active usage time for sessions (session durations with idle time removed)

The duration of sessions varied widely as well, ranging from a few seconds to several hours. The median session length for the community was 9 minutes (IQR 33 minutes, mean 59 minutes, SD 3 hours 49 minutes). However, this measure simply refers to the duration for which the application was open. To get a sense of how much time was spent actively using the application, we reconsider these figures with idle periods removed. Given that the mean time between commands was 19 seconds, we chose 120 seconds as a conservative threshold for idle time. Once idle time is removed, Figure 1 shows that most sessions include very little active usage time. The median active usage time across sessions was 6 minutes (IQR 15 minutes, mean 16 minutes, SD 26 minutes).

On the whole, while there were a small number of users who used ingimp frequently and actively for long periods, most used ingimp infrequently to perform short tasks.

4.4 Characterizing Users' Documents

A total of 13,609 images were operated on during the data collection period. Looking at characteristics of the images, we see that the median maximum image resolution is 800x691 (IQR 1131x1152, mean 1176x989, SD 1408x1008). These results suggest that the primary uses of ingimp do *not* include working with high-resolution bitmap images from digital cameras. Also of interest is the maximum number of layers[1] per image, which provides an indication of the complexity of both the document and the user's task. The median maximum number of layers in an image was 1 (IQR 1, mean 4, SD 13.6). This implies that the majority of ingimp users are not professionals, since professionals tend to work on complex documents and utilize many layers as part of their workflow [19].

5 Community Command Usage

Our data analysis thus far suggests that the majority of ingimp users performed relatively short, targeted tasks on documents of modest complexity. To gain a clearer picture of how participants used the application, we now consider command usage.

ingimp records every command that is placed on the undo stack. It also logs Undo and Redo, the two meta-commands that operate on the undo stack. Across the entire application, there are approximately 500 different commands available to users. These commands include those that modify entire regions at once (such

[1] Modern bitmap editors allow one to define multiple *layers*, where each layer contains a unique bitmap. These layers are combined to create the visible image using compositing operations and masks.

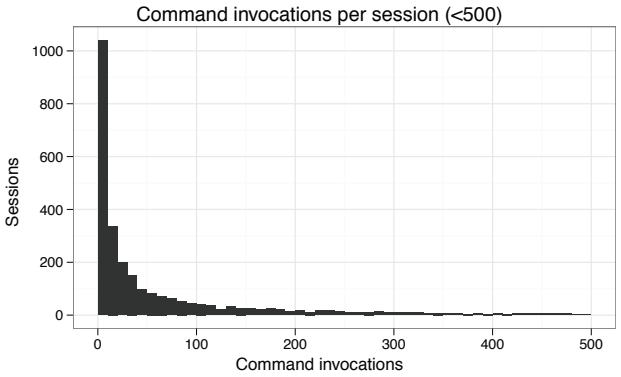

Figure 2. Histogram showing the number of command invocations per session (bins of width 10)

as filters), direct manipulation tools (such as the paintbrush), and those that operate on properties of the document (such as layer compositing operations).

In this section, we characterize the number of commands used in sessions, consider which commands were most commonly used by the community according to a number of metrics, and examine the size of users' command vocabularies. In these analyses, we consider only sessions in which at least one command was logged. This criterion excludes a number of sessions for which no commands were placed on the undo stack (1288, or 31% of all sessions). (While these sessions lack commands appearing on the undo stack, this should not be interpreted as individuals not using the application. An analysis of user-supplied task descriptions revealed that many users report using the application to view images, convert images between file formats, or to obtain RGB color values of pixels. These uses of ingimp do not result in commands being placed on the undo stack, explaining the large percentage of sessions with no logged commands.)

5.1 Command Statistics for Sessions

The number of command invocations per session ranged from 1 to 9236 with a median of 24 (IQR 107, mean 167, SD 480) (see Figure 2). If one counts repeated, successive invocations of the same command as only one command, the median is 14 (IQR 47, mean 57, SD 123, min 1, max 1489). As we discuss in the next section, the main reason for this large disparity is the effect of direct manipulation tools on total command counts. Finally, if we consider only the number of unique commands used in a session, there is a median of just 6 commands per session (IQR 9, mean 9.2, SD 9.6, min 1, max 85).

In conjunction with our previous analysis of active usage time, these numbers strongly indicate that users are engaging in relatively simple, targeted tasks. For more extended or involved creative work, we would expect to see more active usage time, and for more complex tasks we would expect to see more command invocations per session.

Next, we look at the commands that were most commonly used by the community. We first describe how this analysis is more nuanced in applications that make heavy use of direct manipulation as an interaction technique.

5.2 A Note on Counting Commands

Previous work reported command use mainly by counting the total number of command invocations across all log files [4,6,10,12]. However, the applications studied in this prior work made little, if any, use of direct manipulation as an interaction technique.

Command Name	Raw Count	Raw Rank	%due to Repeats	Percent Users	User Rank
Add Anchor	75858	1	96%	21%	44
Undo	71301	2	59%	91%	1
Eraser	60157	3	85%	47%	16
Paintbrush	44300	4	82%	71%	4
Bucket Fill	24591	5	89%	59%	12
Smudge	21119	6	97%	25%	37
Clone	12594	7	95%	18%	51
Pencil	12499	8	81%	39%	23
Rect Select	10721	9	15%	69%	6
Fuzzy Select	9621	10	58%	44%	21
Move Floating Selection	9588	11	28%	70%	5
Item visibility	9234	12	51%	62%	9
Select None	9083	13	9%	78%	2
Move Layer	7812	14	42%	62%	10
Ink	7753	15	83%	11%	87
Paste	7543	16	5%	74%	3
Text	4949	17	68%	46%	20
Add Layer	4873	18	19%	60%	11
Anchor Floating Selection	4463	19	0.1%	63%	8
Airbrush	4192	20	92%	17%	57

Table 1: Command rankings by total number of invocations across the entire community. Shading indicates major points of difference between ranking methods.

As we previously mentioned, ingimp logs a command for each modification to the document. This logging strategy can cause a large variation in the frequency of log entries for different types of commands. For example, direct manipulation tools such as the paintbrush might be disproportionately represented in log counts because each individual paint stroke is recorded as a separate event. Conversely, applying a filter will only result in one logged command, regardless of how long the user spends adjusting settings before it is finally applied. As a result, one cannot simply assume that the commands with the highest number of invocations are the most commonly used commands. Instead, it is necessary to consider command counts from a number of different perspectives to gain a more holistic picture of application use.

Toward this end, we consider command usage in the following ways:

- Raw command counts across the entire community (i.e., the total number of invocations across all users and sessions)
- The number of users who have used a command at least once
- The number of sessions in which a command appears at least once
- Repeated, successive invocations of the same command collapsed to a single invocation of that command

5.3 Commonly Used Commands: Raw, User, and Session Counts

Table 1 shows the top 20 commands ordered according to the total number of invocations across all users and sessions (i.e., raw counts). It also provides the command's ranking in this ordering

Command Name	Percent Users	User Rank	Percent Sessions	Session Rank	Raw Rank
Undo	91%	1	62%	1	2
Select None	78%	2	34%	3	13
Paste	74%	3	34%	4	16
Paintbrush	71%	4	26%	7	4
Move Floating Selection	70%	5	30%	5	11
Rect Select	69%	6	35%	2	9
Scale Image	63%	7	21%	12	29
Anchor Floating Selection	63%	8	25%	8	19
Item visibility	62%	9	24%	10	12
Move Layer	62%	10	24%	9	14
Add Layer	60%	11	28%	6	18
Bucket Fill	59%	12	18%	13	5
Crop Image	57%	13	24%	11	27
Add Text Layer	52%	14	16%	17	31
Select All	51%	15	11%	23	52
Eraser	47%	16	18%	14	3
Redo	47%	17	17%	15	21
Cut	46%	18	14%	18	24
Remove Layer	46%	19	17%	16	28
Text	46%	20	12%	20	17

Table 2: Command rankings by percentage of users who have used the command at least once. Shading indicates major points of difference between ranking methods.

scheme, which is simply the numbers 1–20 in increasing order. We introduce this ranking convention here to assist in understanding other command rankings, discussed momentarily.

The commands with the greatest number of total invocations are Add Anchor, Undo, Eraser, Paintbrush, and Bucket Fill. If we consider what percentage of these invocations are due to repeated, successive invocations of the same command (represented in the "% Due to Repeats" column), we see a large effect due to repeated invocations. In fact, more than half of the commands in this list can attribute their high frequency counts to repeated invocations. For example, 96% of all invocations of Add Anchor are repeated invocations.

The second most frequently invoked command is Undo. Given the relatively high repeat counts of this command (59%), it appears that Undo is partially used to return to a previous state after going down a path that proves to be suboptimal. This type of behavior is what one would expect if users work on ill-defined tasks [19].

Table 1 also provides data on the percentage of users who have used each command at least once, and the command's corresponding rank when ordered by this measure (shown as "user rank"). We have highlighted commands with a user rank greater than 20. The highlighting clearly shows the limitations of considering raw command counts alone when forming a picture of how a community uses applications with direct manipulation tools. For example, Add Anchor, the top ranked command by raw counts, was only used by 21% of users. Similarly, the Smudge, Clone, Ink, and Airbrush tools were used by less than a third of the community. Their prominence in this ordering, then, should not be taken as a sign of their relative importance to the larger community.

To balance the above perspective, we now consider commands according to two other metrics: the user rank, introduced above, and the session rank (the number of sessions in which the command is used at least once). A command's user rank provides an indication of how widespread use of the command is in the community, while the command's session rank indicates how vital it is across all tasks performed by the community.

Table 2 shows the top 20 commands according to the user rank metric. In this ordering of commands, direct manipulation tools are noticeably less prominent. Instead of Add Anchor, Undo appears as the most widely used command, with 91% of users having used Undo at least once. There is a fairly significant jump down to the next command, Select None (78%), followed by a more gradual drop-off. Paintbrush still appears prominently in this list, suggesting that many ingimp users spend at least some time painting or manually modifying pixels in images (e.g., touching up images). In fact, the commands in this list strongly suggest that many users at least occasionally use ingimp for content creation tasks, such as painting or graphic design, evidenced by the presence of Paintbrush, Bucket Fill, Eraser, and Text—the primary tools in ingimp for creating and modifying content. The remaining commands deal with selections and layers. Notably absent from this list are commands used for manipulating photographic images, such as those that alter the brightness, contrast, hue, or saturation of an image. In our higher-level task analysis, however, we do find that basic image correction is an activity performed by a subset of ingimp users.

Table 2 also lists the percentage of sessions in which the commands have been used, along with the command's session rank. In comparing these metrics, we see there is a fairly good correspondence between the percentage of users using a command at least once and the number of sessions in which it has appeared. In fact, there is only one command that is present in the user rank top 20, but not the session rank top 20 (Select All), and its session rank is 23.

5.4 Command Coverage and Command Vocabularies

Across all sessions, there were a total of 487,308 command invocations of 352 different commands. Since there are approximately 500 different commands available in ingimp, the user community is not exercising all of the available functionality. This finding is consistent with the findings of Hanson *et al*. [7] for the UNIX command line and for Linton *et al*.'s analysis of MS Word in which only 152 of 642 commands were used [12].

In terms of users' command vocabularies (the set of commands the user has been observed using [6]), we find that the size of an individual user's command vocabulary ranged from 1 to 169, with a median of 27 (IQR 29, mean 34, SD 27.9), or less than 6% of the total number of avaliable commands. This finding, that users' command vocabularies tend to be small in comparison to the number of available commands, is consistent with previous findings [6,12].

Considering the above observation that users' command vocabularies tend to be small, along with the previous observation that users tend to use only a small number of unique commands in a given session, we see that ingimp offers far more functionality than is effectively utilized by most of its user population, particularly for any given session.

5.4.1 Overlap in Command Vocabularies

The sparse use of the application's functionality raises the question of how much overlap there is between users' command vocabularies. That is, do many users share a few small sets of

Proportion of users	Commands	% Commands
90–100%	1	0.3%
80–90%	0	0%
70–80%	4	1.1%
60–70%	5	1.4%
50–60%	5	1.4%
40–50%	7	2.0%
30–40%	6	1.7%
20–30%	17	4.8%
10–20%	50	14.2%
0–10%	257	73.0%

Table 3. The number of observed commands shared by different proportions of users

commands? To explore this concept, we consider the number of commands shared by different proportions of the user community.

Table 3 shows that there is very little overlap in command vocabularies across the entire community. There was no single command used by all users, though Undo was used by 91% of users. In fact, only 15 commands were used by greater than half of the population. Futhermore, 257 commands, representing 73% of the total number of distinct commands observed, were used by no more than 10% of the user community. We can conclude that users' command vocabularies are fairly distinct, indicating either that ingimp is used for widely varying tasks by different users or that users have differing methods for achieving similar goals. This finding closely mirrors those of Greenberg [6], and Sutcliffle and Old [17] for the UNIX command line domain.

6 Identifying Higher-Level Tasks

The frequency counts of commands explored in the previous section provide perspectives on which commands are commonly used across the entire community, but they do not say much about the higher-level tasks performed by users, or which commands are commonly used together.

One way to understand what tasks are performed by the community is to identify what sets of commands are frequently used together. However, this is a not a trivial task. While it is relatively easy to generate transitional probabilities from one command to another, previous work argues that this unit of analysis does not easily extend to describing higher-level tasks. For example, Greenberg [6], and Sutcliffe and Old [17] found that this approach leads to fragile models because users' command vocabularies tend to be small and idiosyncratic—a result that we have found for ingimp users as well.

In this section, we demonstrate how clustering of sessions can yield insight into the types of tasks being performed by the community, and identify commands that are frequently used together. The result of applying this approach to our dataset is compelling as the sets of commands it produces suggest specific tasks. Further, we provide evidence for the validity of the results by examining the activity tags (the optional text-based task descriptions users could enter at application start-up) associated with sessions in the resultant clusters. We begin by describing the method used to cluster sessions.

6.1 Clustering Sessions

To determine common tasks, we take the following approach. We start by collecting sequences of commands that correspond to separate (though unknown) tasks. We then cluster these sequences based on some measure of similarity. Finally, we extract the frequently occurring commands from the resulting clusters of sequences.

The first step, partitioning command sequences by task, would be difficult if users performed multiple tasks per session. However, our previous analyses have demonstrated that most sessions are relatively short, have few command invocations, and include few documents. These trends suggest that individual sessions are a reasonable approximation of individual tasks. Accordingly, we apply our clustering technique on the granularity of sessions, to all sessions in which at least one command was logged (2906 sessions in total).

To perform the clustering itself, we adapted a clustering approach of Whissell *et al*., previously used to characterize document similarity [20]. We first create a feature vector containing the command counts for each session. These vectors are then input into 11 well-known clustering algorithms (e.g., *k*-means), each of which outputs 7 clusters of sessions. From these 11 sets of 7 clusters, we would like to identify the "best" set of clusters. However, there is no objective function for making this determination. To address this, the Whissell approach works as follows. For each set of clusters produced, the clustered sessions are used as labeled training data to train a classifier for classifying sessions into clusters. The accuracy of each classifier is determined using ten-fold cross-validation. To address the problem of a trivial classifier that places everything in the largest cluster, the accuracy score is normalized by the number of sessions in the largest cluster. The set of clusters with the highest final accuracy score is considered the best set of clusters.

From each of the resultant clusters of sessions, we identify the 12 most frequent commands in each cluster (using the session rank metric) to create seven different *task sets*—sets of commands that typically appear with one another. Because we are clustering sessions, a single command may appear in multiple task sets. However, this is a desirable property since different tasks naturally share commands.

Table 4 shows the seven task sets produced by applying this technique to the ingimp dataset. Inspection of the task sets suggests particular user activities. For example, the first cluster suggests fairly basic photo manipulation tasks, such as rotating, resizing and scaling. It also includes commands typically used for photo retouching, such as Levels (used to adjust an image's brightness and color balance), and Unsharp Mask, a filter used to sharpen images. The second cluster's task set suggests a particular operation: pasting an image into the current document from the clipboard (which results in a floating selection), choosing a place for the pasted image (Move Floating Selection), and then anchoring it to the page (Anchor Floating Selection). In the next section we find that user-supplied activity tags provide a more detailed picture of what users are doing in this cluster.

The third cluster's task set suggests working with text in an image (evidenced by Text, Add Text Layer, and Move Text Layer), and the fourth cluster's task set suggests use of the Paths tool (evidenced by Add Path, Add Anchor, and Drag Anchor). The fifth through seventh clusters collectively suggest painting and graphic design tasks, though more specific activities are less clear.

6.2 Activity Tag Keywords for Clustered Sessions

To validate the effectiveness of the clustering, we examined users' activity tags. Example tags entered by users include "photo manipulation", "screenshot editing", "resizing", and "Logo creation".

In total, 608 of the 2906 clustered sessions included activity tags. To summarize these activity tags, we broke them into individual keywords and counted keyword occurrences for each cluster. Since many keywords occur in different tenses (e.g., "resize" and "resizing") or singular and plural ("screenshot" and "screenshots") we manually stemmed all keywords to their simple, non-plural present tenses (e.g., "crops" and "cropping" both become "crop"). We also filtered out common stop words such as "for",

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Scale Image	Rect Select	Add Text Layer	Undo	Undo	Undo	Undo
Crop Image	Select None	Undo	Item Visibility	Bucket Fill	Eraser	Paintbrush
Undo	Undo	Move Layer	Add Layer	Select None	Move Floating Selection	Paste
Levels	Paste	Text	Paste	Rect Select	Fuzzy Select	Add Layer
Resize Image	Move Floating Selection	Move Text Layer	Add Path	Move Floating Selection	Select None	Select None
Rect Select	Anchor Floating Selection	Add Layer	Add Anchor	Paste	Add Layer	Move Floating Selection
Select None	Crop Image	Paste	Move Layer	Anchor Floating Selection	Paste	Anchor Floating Selection
Convert Image to Grayscale	Cut	Remove Layer	Select None	Clone	Anchor Floating Selection	Redo
Rotate Image	Move Layer	Select None	Remove Layer	Item Visibility	Paintbrush	Rect Select
Paste	Add Layer	Rect Select	Set Preserve Trans	Add Layer	Rect Select	Item Visibility
Rotate	Item Visibility	Item Visibility	Move Floating Selection	Move Layer	Move Layer	Move Layer
Unsharp Mask	Scale Image	Anchor Floating Selection	Drag Anchor	Paintbrush	Item Visibility	Eraser

Table 4. Task sets for the seven clusters of sessions

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
(28) resize	(29) screenshot	(6) logo	(8) design	(7) design	(7) create	(11) design
(15) photo	(15) edit	(5) photo	(7) web	(5) correction	(5) correction	(7) graphic
(13) crop	(12) web	(4) text	(6) graphic	(4) graphic	(4) web	(7) draw
(10) web	(11) crop	(4) make	(6) edit	(3) edit	(4) graphic	(6) test
(9) screenshot	(10) resize	(4) edit	(5) texture	(3) create	(4) edit	(5) texture
(9) edit	(6) photo	(3) design	(4) make	(2) web	(4) design	(5) resize
(7) scan	(6) mockup	(3) create	(4) gif	(2) test	(4) creation	(5) create
(6) screen	(6) design	(2) web	(4) correction	(2) retouch	(4) background	(4) edit
(6) correction	(6) cut	(2) up	(3) png	(2) out	(3) photo	(4) correction
(5) quick	(5) up	(2) test	(3) photo	(2) crop	(3) icon	(3) work

Table 5. Frequently occurring activity tag keywords for each cluster (occurrence counts in parentheses)

"and", and "the", as well as "image", which occurred frequently across all clusters and does not indicate any particular activity. The ten most frequently occurring keywords for each cluster are shown in Table 5.

Our first observation is that, while some keywords such as "photo", "web", "crop", and "correction" occur across clusters, the top keywords for each cluster are relatively distinct. Moreover, keywords that do occur across clusters often occur more prominently in one particular cluster (e.g., "resize" in the first cluster and "screenshot" in the second).

As well as being distinct, the keywords appear to fit with our intuitions on the user activities represented by each cluster. For example, cluster 1's task set includes commands for photo retouching, and it features activity tag keywords such as "photo", "resize", "crop", and "edit".

The top keywords for cluster 2 also match our interpretation from the task set (pasting an image from the clipboard and then performing some operation on it) and tell a richer story. While cutting and pasting is not mentioned directly, the most common keyword is "screenshot". This observation is significant because a common method of taking a screenshot in Windows is to use the "Print Screen" key on the keyboard, which copies an image of the screen to the clipboard so it may be pasted into an application. In fact, one activity tag for a session in this cluster was "pasting a screenshot".

Though there are fewer total keywords in cluster 3 than in other clusters, "logo", "text", and "design" are featured prominently, which match with the text operations suggested by the cluster's task set in the previous section.

Finally, clusters 5, 6, and 7, whose task sets included direct manipulation tools and suggested painting and graphic design activities, include keywords such as "design", "draw", "graphic" and "create". The keyword "correction" is also featured prominently, which fits with direct manipulation tools such as Paintbrush, Eraser, and Clone (a tool often used to fix small blemishes in images).

In sum, the resultant task sets both appeal to intuition and show good correspondence with users' reported intentions. More importantly, they provide a richer picture of the activities performed by an entire community of users than command counts alone.

7 Discussion

In this work, we have introduced new perspectives for understanding long-term application use, and replicated a number of findings from previous work for a new application context. The replication of past findings is important for two reasons. First, few such studies exist in the literature, and second, we examined use patterns of an application with a qualitatively different style of interaction (namely, heavy use of direct manipulation). As such, this work helps to generalize the results of past work.

Arguably, the most important replication of past findings concerns the relatively limited use of available application functionality. Because applications are designed to accommodate diverse

populations with varying tasks, they are becoming extremely packed with functionality. Yet, our study, like those of Hanson *et al*. [7], Greenberg [6], and Linton *et al*. [12], show that individual users require only a very small subset of this available functionality. This observation has implications for interface design, as we describe.

While there are certainly advantages to large monolithic applications (e.g., they can be marketed to a wide user base and can accommodate users' changing needs), application designers must consider the impact of rich feature sets on usability. More specifically, prior research indicates that excess interface complexity has both quantitative and qualitative impacts on user experience. Quantitatively, the number of interface elements negatively impacts task time, particularly for more novices users, whose visual search time is a linear function of the number of "relevant" items present (e.g., within a given menu or toolbar) [2]. As expertise increases, the impact of excess complexity on performance is lessened; however, for all but perhaps the most expert users, command selection time is always some function of the number of relevant commands available [2,15]. Qualitatively, many users, irrespective of their expertise, find large applications to be overwhelming, frustrating and difficult to navigate [13].

The difficulties users experience with large, complex applications, combined with the limited overlap in their command vocabularies (another important result replicated in our study), indicate that more emphasis should be placed on providing personalization capabilities in interface designs. Outside of research prototypes (e.g., [1,13]), personalization appears to be an afterthought rather than a first class-interface object, and is often tucked away in menu structures or offered in ways that are cumbersome to use. As an example, in the Microsoft Ribbon, only the very top toolbar can be personalized, and adding anything to this toolbar requires picking and choosing individual commands from long lists (such as an alphabetized list of all commands). This type of personalization is likely to be difficult for novices, who might not know which commands are relevant to their tasks.

One of the new methodologies that we have presented is the automatic clustering of sessions to determine which commands frequently appear together in an individual session. An obvious application of this technique is to better support personalization. By identifying groups of commands commonly used together, interface designers could enable coarse-grained personalization, in which users select groups of functionality rather than individual commands. In addition, if these clusters have higher-level interpretations, as was the case for our clusters, they could be labeled with intuitive names (e.g., "painting tools" or "image correction tools"). Such labeling could facilitate personalization by novices, who are likely to have some notion of their tasks, even if they don't know the specific commands they will need.

Finally, in our work, the application of a clustering technique to tasks was facilitated by the fact that individual sessions served as a reasonable approximation of individual tasks. This may not be the case for other application domains. Consider, for instance, an email application. Many users will open such an application once in the morning, and perform various tasks with it throughout the day. In cases such as these, the clustering technique that we have presented is still applicable, but a different measure of what constitutes a task must be identified. In the email example above, clustering could be applied based on sequences of commands applied with little idle time between them, or on sequences of commands applied to particular email messages.

8 Conclusion and Future Work

In this paper, we have added to the body of research describing large-scale, long-term use of feature-rich desktop applications by considering patterns of use of an application that makes heavy use of direct manipulation as an interaction technique. We have also introduced additional analytical perspectives for summarizing data from such applications. Finally, we have introduced a novel method of gaining a higher-level understanding of the types of tasks a community performs based on command usage alone.

Our data motivates the need for continued work in user interface personalization. In this respect, one promising area is the application of task-based clustering techniques to help identify sets of commands commonly used together by a user base. These sets of commands would then enable high-level, task-centric personalization interfaces.

References

[1] Bunt, A., Conati, C., and McGrenere, J. Supporting interface customization using a mixed-initiative approach. *Proc. of IUI 2007*, ACM (2007), pages 92–101.

[2] Cockburn, A., Gutwin, C., and Greenberg, S. A predictive model of menu performance. *Proc. of CHI 2007*, ACM (2007), pages 627–636.

[3] Cook, R., Kay, J., Ryan, G., and Thomas, R.C. A toolkit for appraising the long term usability of a text editor. *Software Quality Journal 4*, 2 (1995), pages 131–154.

[4] Draper, S.W. The Nature of Expertise in UNIX. *Proc. of INTERACT '84*, Elsevier North-Holland (1984), pages 465–471.

[5] Ducheneaut, N. and Moore, R.J. The social side of gaming: a study of interaction patterns in a massively multiplayer online game. *Proc. of CSCW 2004*, (2004), pages 360-369.

[6] Greenberg, S. *The computer user as toolsmith: the use, reuse, and organization of computer-based tools*. Cambridge University Press, New York, NY, USA, 1993.

[7] Hanson, S.J., Kraut, R.E., and Farber, J.M. Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Information Systems 2*, 1 (1984), pages 42–57.

[8] Heer, J. and Chi, E.H. Separating the swarm: categorization methods for user sessions on the web. Proceedings of *CHI 2002*, ACM (2002), pages 243–250.

[9] Hilbert and Redmiles. Extracting usability information from user interface events. *ACM Comp. Surveys 32*, 4 (2000), pages 384–421.

[10] Kay, J. and Thomas, R.C. Studying long-term system use. *Communications of the ACM 38*, 7 (1995), pages 61–69.

[11] Kosala, R. and Blockeel, H. Web mining research: a survey. *SIGKDD Explor. Newsl. 2*, 1 (2000), pages 1–15.

[12] Linton, Joy, and Schaefer. Building user and expert models by long-term observation of application usage. *Proc. of UM 1999*, Springer-Verlag New York, Inc. (1999), pages 129–138.

[13] McGrenere, J. and Moore, G. Are we all in the same "bloat"? *Proc. of GI 2000*, (2000), pages 187-196.

[14] Murphy, G.C., Kersten, M., and Findlater, L. How Are Java Software Developers Using the Eclipse IDE? *IEEE Software 23*, 4 (2006), pages 76–83.

[15] Norman, K.L. *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*. Greenwood Publishing Group Inc., Westport, CT, USA, 1991.

[16] Obendorf, H., Weinreich, H., Herder, E., and Mayer, M. Web page visitation revisited: implications of a long-term click-stream study of browser usage. *Proc. of CHI 2007*, (2007), pages 597-606.

[17] Sutcliffe and Old. Do users know they have user models? Some experiences in the practice of user modelling. *Proc. of INTERACT '87*, Elsevier North-Holland (1987), pages 35–41.

[18] Terry, M., Kay, M., Van Vugt, B., Slack, O., and Park, T. ingimp: introducing instrumentation to an end-user open source application. *Proc. of CHI 2008*, ACM (2008), pages 607–616.

[19] Terry, M. and Mynatt, E.D. Recognizing Creative Needs in User Interface Design. *Proc. of the Fourth Conference on Creativity & Cognition*, ACM Press (2002), pages 38–44.

[20] Whissell, J.S., Clarke, C.L.A., and Ashkan, A. Clustering web queries. *Proc. of CIKM 2009*, (2009).

[21] Whiteside, J., Archer, N., Wixon, D., and Good, M. How do people really use text editors? *ACM SIGOA Newsletter 3*, 1–2 (1982), pages 29–40.