The CLOTHO Project: Predicting Application Utility

Joshua Hailpern
Department of Computer Science
University of Illinois
jhailpe@cs.uiuc.edu

Karrie Karahalios
Department of Computer Science
University of Illinois
kkarahal@cs.uiuc.edu

ABSTRACT
When using the computer, each user has some notion that "these applications are important" at a given point in time. We term this subset of applications that the user values as *high-utility applications*. Identifying *high-utility applications* is a critical first step for Task Analysis, Time Management/Workflow analysis, and Interruption research. However, existing techniques fail to identify at least 57% of these applications. Our work directly associates measurable computer interaction (CPU consumption, window area, etc.) with the user's perceived application utility without identifying task. In this paper, we present an objective utility function that accurately predicts the user's subjective impressions of application importance, improving existing techniques by 53%. This model of computer usage is based upon 321 hours of real-world data from 22 users (both professional and academic). Unlike existing approaches, our model is not limited by a pre-existing set of applications or known tasks. We conclude with a discussion of the direct implications for improving accuracy in the fields of interruptions, task analysis, and time management systems.

Author Keywords
Application Utility, Application Importance, Modeling

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION AND MOTIVATION
Knowing which applications are "important" to a user at a given point in time is a subjective evaluation. Yet research in multiple domains of HCI must make use of an importance metric to facilitate their analysis and predictions.

Therefore the quality of the analysis is necessarily dependent on an accurately calculated importance metric. Consequently, a poorly calculated importance metric will degrade the analysis, just as a malfunctioning thermometer will degrade the quality of a scientific experiment.

Consider Task Analysis Research [10,16,18,21,26], which attempts to group relevant applications based on the goal/task they are part of. This Task-based classification relies upon knowing, a priori, which applications are important and which are peripheral. Similarly, Interruption Research [6,14,15,24] focuses on identifying when users switch between activities, often basing these assessments on accurately knowing when applications move in and out of favor. Time Management/Workflow analysis [2,23,30] likewise is based on an initial assessment of salient applications before applying their logic and models. What users describe as "important" applications are the same as the relevant/in favor/salient applications on which these high-level analyses are based.

While all of these contextual research areas assume an accurate importance metric before conducting their desired analysis, in reality they naively minimize this critical first step of assessing important applications by using unsubstantiated metrics. Most research utilize window focus or window switches to divine which applications are important, yet our real world data shows that these techniques have at least a 57% failure rate. Without an accurate method for predicting these important applications, even the best contextually sensitive software solutions will be handicapped. Therefore, our goal is to take the guess work out of predicting important applications and replace it with a concrete well justified model.

We hypothesize that there is a quantitative metric to accurately determine importance that we term ***high-utility applications.*** This paper aims to bring together a user's perceived application utility and a system's measured resource allocation (CPU, RAM, screen real-estate, and other system resources) resulting in a 53% improvement in the predication of *high-utility applications* over existing techniques. We believe that the aforementioned research would benefit from our improved prediction of *high-utility applications*.

Consider a concrete example where a graphic designer is working on a large client presentation. This user might have a presentation development application open and in focus,

while also busily entering text. Meanwhile adjacent to the presentation software, the designer might also have a PDF viewer open for reference (not in focus, but mostly visible), which the user glances at periodically. Though partially obscured, a web browser pointed at a social networking site, might also sit on the desktop, while a media player plays music in the background. If asked to recall which of these applications were related to presentation design, the user could easily state "presentation software and PDF viewer."

Our goal was to create a model that would predict that the presentation software *and* PDF viewer were *high-utility applications*, while the media player and web browser were not (even though they were active and may have even occasionally received focus). In other words, the contribution of this paper is a model that accurately predicts a user's impression of application utility (based on a large and diverse data set) without task and interruption recognition logic, which is highly transient and often has limited scope. By using generalized application interactions and resource allocation, we hope to cross domains of computer use, and have a more robust model for less common and new applications/use-patterns. Further, a task/goal independent model will be resilient to multi-tasking scenarios because it assess each application and assigns it a utility value regardless of use in multiple tasks or concurrent/competing goals. This is not Task Analysis, Workflow Analysis, or Interruption research, rather a model to enhance and improve these other techniques.

We begin by reviewing existing and related approaches to detecting/measuring application utility and computer usage. Based on prior research, we outline our research question, detail our methodology, and present our set of predictive variables. Then we discuss the construction of our models, their 53% improvement on existing techniques (0.62 Kappa score), and subsequent limitations. We conclude with a discussion of applications and future work.

RELATED WORK

We describe other high level prediction systems, which have gathered empirical aspects of computer usage in order to link them with subjective metrics. These systems are in the areas of Interruption Identification [6,14,15,24], Goal/ Task Identification [10,18,21,26], Task Resumption [16], Workflow Analysis[2], Web Revisitation patterns[1], and general computer use patterns [28]. We then detail how our work builds on their approaches.

Predicting Application Utility

The nature of most modern computer systems (high CPU, RAM, screen real-estate, and other system resource availability) allows many concurrent applications to be open at once, regardless of whether a given application is being actively used or not. This has led to multiple areas of HCI research that focus on identifying/predicting high-level activities such as predicting user's goals, tasks, or identifying workflows [1-3,7-10,12,16,18-24,26-30,35,36]. Most of these research areas base their models and recognition/ identification logic on knowing which applications are high utility or "in use." Yet the identification of *high-utility applications* is confounded by the high degree of system "background noise" generated by non-relevant applications [12]. Oliver et al. [26] suggest that at least 20-30% of open windows may be spurious, or unrelated to current activity.

We hypothesize that there is a generic quantitative metric to accurately measure application utility, thus directly assessing *high-utility applications* and those that are "background noise." By avoiding application-specific interactions, such as webpage reloads [1,9,18], an application utility prediction metric can be robust against a broad and ever-changing field of applications. Similarly, Grudin suggests that failures in usability research are often due to failures in understanding context [11]. Therefore, by creating a metric which does not utilize task/goals in the evaluation process, we will not be hindered by behavior and use patterns that are constantly changing, in a limited and predetermined set, or not fully understood by the prediction system.

The suppositions by Hilbert and Redmiles [13], state that a wide spectrum of UI interactions are needed to infer higher-level modeling. However, the question is open as to which metrics should be used. The following sections discusses existing practice.

Choosing Predictive Measures

Despite the successes in the large corpus of related literature, sparse evidence is presented (except for subjective observations without scientific validation) to justify the selection of measures for predicting/identifying forms of interaction. Further, selected variables are not uniform across existing literature. Some work utilizes window focus [9,12,26,28,33], mouse/keyboard activity [14,28], text in a window [10,17], window switches [26], CPU load, page loads [28], and file I/O [12] with little overlap. Kelly and Belkin [19], in their study of web visitation, suggest that dwell time for websites may not be the sole indicator of task/focus. Extrapolating from their findings may suggest that window focus may not be the sole factor in determining application relevance (though an intuitive place to start), especially with displays and computers supporting multiple applications simultaneously. Without experimental evidence to support variable selection, choosing dependent measures is a guessing game.

To this end, we seek to quantitatively assess which features of computer usage are indicative of application utility. Our approach builds upon Hilbert and Redmiles [13], who advocate using a wide spectrum of UI interactions to infer higher-level modeling. While their work does not indicate which specific variables are the most predictive, their success indicates the utility of a multiple variable approach. We build upon previously used variables to determine predictive value. Further, few studies examine behavior over time. One notable exception is [17], which studied behavior over 300 and 600 seconds intervals. The selection of these

time intervals was not based on existing literature. Our work draws on the results of Iqbal and Bailey[14] and their calculation of task breakpoint intervals to guide our analysis intervals.

RESEARCH QUESTION

We introduce the following research question:

> Is there a general utility function based on measurable aspects of computer usage (e.g., CPU, window size, etc.) that accurately predicts the user's impression of application utility?

In other words, we aim to link actual system-level activity with subjective application utility to create a general predictive model of application value. To achieve this end, we conducted a experience sampling study, whose data was used to build predictive models.

METHODS

We recruited 36 users to participate in a week-long (5 day), real-world, data collection process. Of the 36 users, 22 (61%) agreed to participate, completed the process, and returned data. Our goal was to link artifacts of computer usage with perceived application utility at a given point in time. To achieve this, we designed, built, and distributed the CLOTHO (**C**omputer **L**ogistical **O**perations and **T**emporal **H**uman **O**bservation) system. CLOTHO allowed us to collect computer resource allocation and UI interactions (*predictive variables),* and link them with human generated data (*dependent variable)*. We can then make predictive models using our sets of predictive and dependent measures. Using CLOTHO, we collected 321 hours of computer and human data over a total of 126 user-days (resulting in 2,294 sets of data points). CLOTHO was built in Cocoa and run on Apple Macintoshes with OS X 10.5+.

Activity-Debriefing Association (Dependent Variables)

To capture perceived application utility over the experimental period, we utilized recall-based experience sampling data collection [5]. Memory, or recall, is often more important than reality [25]. Therefore, we believe that modeling a user's recalled application utility would provide a more usable model for real-world applications. To this end, our data collection had two steps; prompt generation and nightly debriefing.

Step 1: Prompt Generation

CLOTHO would periodically prompt users for their current activity or goal. Figure 1 presents the user prompt. The text-cursor/focus would jump to the text entry field (with auto fill-in based on prior entries). Participants would enter their description. Upon pressing the return key, participants would resume their current activity and cursor focus. When data was returned to researchers, the specified activity was *not* included. This allowed participants to be detailed about what they are doing without concern about privacy or confidentiality: the description functioned only as a personal memory marker for recall.

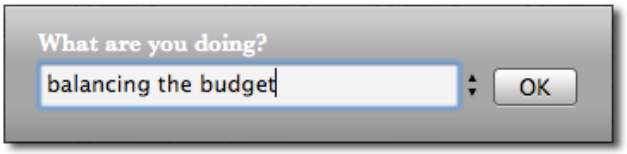

Figure 1. Current Activity Prompt
Users would be alerted by brief sound and dimming of entire screen. Prompt window would then appear in center of the screen.

To ground the prompt interval selections in existing literature, timings were based on average, coarse, and medium breakpoint intervals in [14]. Conceptually, breakpoints occur at conceptual shifts in workflow (changing activity and switching documents, respectively), and might serve as examples of events that are easy to recall for the debrief task (described below). These interval timings, are relatively short and average an interruption every 4 minutes (min). Disruption of a user's regular activity is a known problem with experience sampling. To reduce disruption on users, we randomly selected multiples of the breakpoint intervals (e.g., 3 medium breakpoints away) increasing the time between interruptions. Further, our break point selection was weighted towards intervals between 8-12 min., with an expected average across all prompts at ~10 min. (actual mean 8.4 min.).

Step 2: Daily Debriefing

At the end of each day, users ran a debriefing program which randomly selected 20 activities from the past day and 8 activities from each prior day (without replacement). A description was presented to the user along with a timestamp and an icon grid of all current open applications

Figure 2. User Debriefing Window
Users are presented their recorded activity, date, time and all active applications. Users select all related applications (in this example Calculator and Quicken 2007 are selected).

(Figure 2). Participants were asked to select all applications (binary for each application: yes/no) that were directly related to the specified activity and time by clicking the icon/name. This allowed us to filter out background applications and the "noise" that they produce while active [12]. While a spectrum based approach ("rank how important this application is" using a Likert scale) would yield a more detailed data set, articulating and ensuring all users understood the different degrees of application utility would be a difficult task. Moreover, if a binary response indeed answers our research question, then it would allow us to proceed with this simpler solution. Our goal was to try a simplistic binary method first, and fall back on more complex dependent variable should this system fail.

Placing a debriefing task at the end of each day served to reduced the disruption on users during their work hours.

Raw Computer Data Collection

CLOTHO actively collected participants' system usage and interactions for 5 work days. Given the complexity of HCI [12,31,35], and the findings of Hilbert and Redmiles [13], we elected to collect many measures of computer interaction. We leveraged the metrics used in existing literature, as well as the full extent of detectable computer events on OS X that would not impact system performance for the user. Three main sets of features were collected: *Process Events, System Snapshots* and *Mouse Position*.

Process Events occur when a specific momentary event happens (e.g., application launch/close, switching window focus, waking from sleep, logging in/out of the desktop, and system shut down). When one of these events occur, the originating application, the related window (if applicable), and a date/time-stamp are logged.

System Snapshots are periodic logs of the current state of the system. These snapshots record (at an application and system level) CPU usage, RAM usage (total physical memory used by a process not including shared RAM), window order (z-buffer), desktop size, window size, window coordinates, number of monitors and positions, and whether a window is visible, in another space, or hidden/minimized. Performing a *System Snapshot* requires a relatively larger amount of disk space than *Process Events*. We therefore sampled every 114 seconds (sec.), or "the smallest meaningful and natural unit of execution ends and the next one begins" [14]. System Snapshots also occurred when applications are launched and closed. *Mouse position* (x and y coordinates) were periodically logged every 114 sec. as well.

Data Collection & Privacy

Privacy and impact on the user's day-to-day computer use were top-most concerns when building CLOTHO. Data collection was restricted to aspects of computer usage that could be gathered with little to no impact on system performance. For example, file I/O could be detected, but would be computationally and resource intensive. In addition, many applications routinely write information to disk. Without application-specific hooks and/or user requested I/

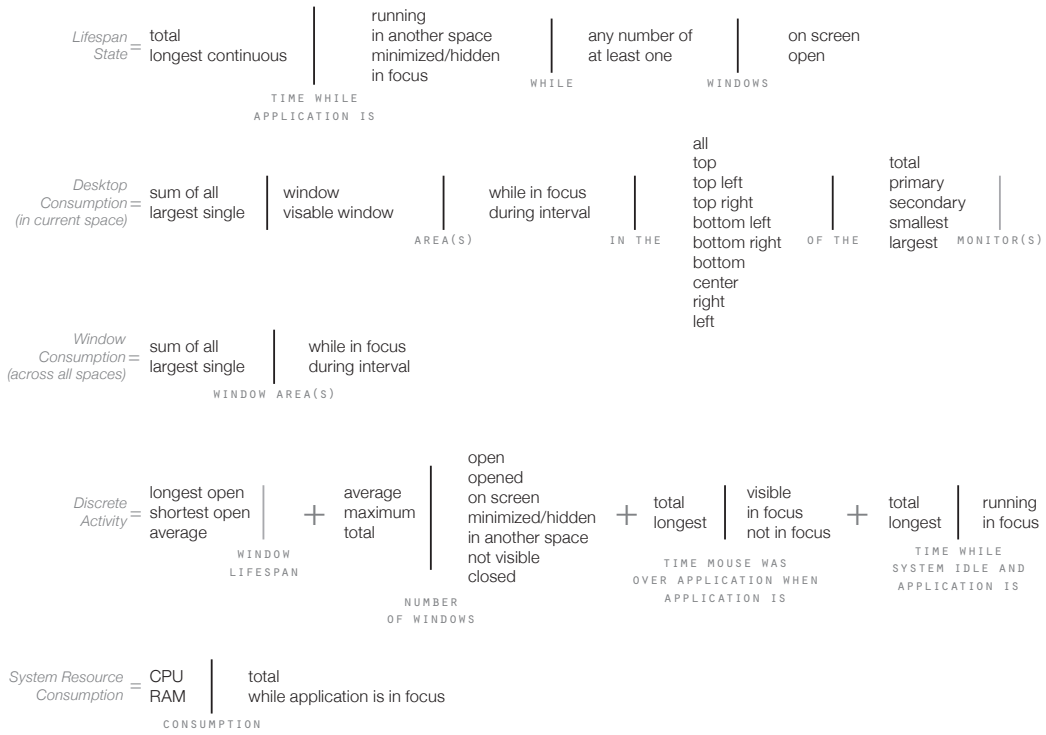

Figure 3. Data Type Predictive Variable Definitions

A visual representation of the 474 "Data Types" that are formed by every permutation separated by the vertical lines. Fore example, one predictive variable from the System Resource Consumption data type is a combinations of (CPU or RAM) and (Total or While in Focus) resulting in "CPU consumption total" or "RAM consumption while application is in focus"

O, routine application disk writes and meaningful disk activity could not easily be differentiated. A single week's participation required approximately 1 GB of disk space. Furthermore, no private/personal/identifiable information was recorded. All data collected were stored in plain text to allow users to verify their privacy. Users' prompts were not returned to researchers ensuring user's would feel free to report what they were doing.

Predictive Variables

For each application presented to the user during a debrief, 11,899 *predictive variables* were calculated and linked with whether or not it was part of the user's activity (*dependent measure*) as marked by the user.

While raw data collected by CLOTHO consists of only 19 features, we calculated a larger set of 11,899 predictive variables. These variables capture not only the raw resource allocation (e.g., size of windows) but a thorough examination of more nuanced aspects of computer usage (e.g., percent of window area in the bottom half of the screen). In addition, we considered the evolution of each resource being consumed over time, to take into account the change, short and long-term consumption.

We determined that there are 474 types of data (Figure 3) that can be extracted from the raw data collected. Each *Data Type* is compared to 7 *Baselines*, and calculated over 5 *Time Intervals*. The permutations of Data Type, Baseline and Time Intervals result in the majority of the 11,899 *predictive variables* (27 other variables were also calculated reflecting the computer's state). The remainder of this section details the calculation of the 11,899 *predicative variables*.

Data Type

Figure 3 illustrates the 474 data types calculated. Each data type has a unique definition and examines user interaction with the computer from a different perspective; System Resource Consumption, Window Consumption, Desktop Consumption, Lifespan State, and Discrete Activity.

Baseline

As suggested by Fenstermacher and Ginsburg[10], data collected should be examined across applications rather than just against the system resources themselves. Therefore, each data type was examined relative to 4 baselines; *Total System Resources, All System Processes, All Applications,* and *Top 3 Applications*. This comparison based approach ensures that all predictive variables are percentages rather than raw values, allowing comparisons to be draw between systems with different configurations. For example, "CPU consumption of Photoshop" divided by "total CPU consumption by all running applications."

Total System Resources: At a basic level, relative *to Total System Resources* seeks to compare an application's resource consumption to the maximum the system can support. For example, "*% window area of an application vs. system desktop*" or "*% CPU usage out of the total CPU power the system can produce*". This dimension presents a raw usage value of each variable relative to the most the system itself can produce.

All System Processes[1]: [10] suggest that we must look across running processes (user and system) applications. For example if application Alpha is using 30% of the CPU, and only 70% of the CPU is being used in total, Alpha would have a value of 30/70 or 43%.

All Applications: To gain a comparison against user driven processes (applications), we also examine each data type against the total resources being used by all (user) applications. For example, screen area would be relative to how much screen area is being covered by all visible applications.

Top 3 Applications: Extending Fenstermacher and Ginsburgh [10], we aim to rank resource consumption across the top 3 applications that dominate each resource. Using relative percentages, ranking can be inferred between running applications. For example, if application A is consuming 20% CPU, B 40%, C 5%, and D 15%, then A vs. the top application would A/B or 50%, indicating that A is using half the CPU power of the top-most application. However, compared to the second most CPU intensive process (A/A), A would have a 100% value, indicating it is the 2nd most CPU intensive application. Likewise, it would have a value of 133% against the third most CPU intensive application.

Time Interval:

Most predictive variables examine the resources used over a period of time. Determining the right interval is essential. Behavior must be examined both in the short and long term use [13] to build a longitudinal model relative to shifts in workflow. Therefore, each predictive variable was calculated at multiples of the fine breakpoint interval (114 sec.) up to one standard deviation away from the coarse breakpoint average (570 sec.) totaling 5 time intervals. This allowed us to examine each behavior from the shortest conceptual shift to the largest.

State Variables:

In addition to the variables generated by the permutations of *Data Type*, *Baseline*, and *Time Interval*, each application was also associated with 27 variables representing the state of the system. They included percentage time (during each time interval) that the system was asleep, idle, off, recording no data (asleep + off), and average number of monitors. We also recored the number of applications running and whether or not each application is in focus at the time of the prompt.

[1] "All System Process" was computed for System Resource consumption only, since conceptually it does not apply to the other categories.

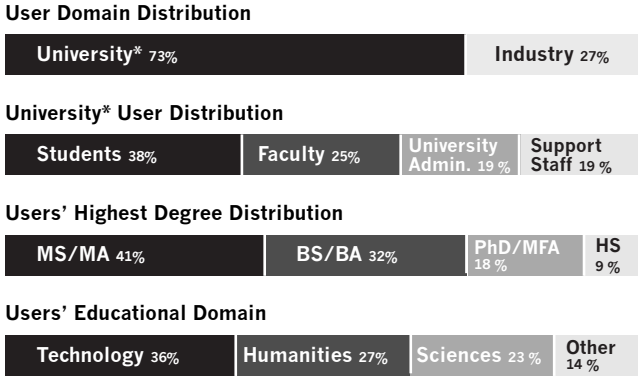

Figure 4. Demographics
**denotes an expanded breakdown of University users*

Participants

Subjects were asked demographic and background information. The 22 participants were recruited from 6 companies and 5 universities with educational backgrounds in more than 17 different areas (Figure 4). Our sample consisted of 55% men with an age range of 21 to 59 (median 46).

Analytical & Statistical Methods

Given the large feature set associated with each dependent variable, we modeled application utility using machine learning, similar to the approach of Bruegge et al., and Dragunov et al.[2,9], but with a different end goal and data sets.

While we collected debrief data from the past day and from each prior day, this paper focuses on debriefed data from the "past 24 hours." Thus, our models were trained and tested on 16,591 data points each of which was associated with 11,899 predictive variables. 3,002 data points were marked as *high-utility applications* (18.09%).

We examined three modeling approaches to create a utility function: *Naive Bayes*, *Logistic Regression*, and *Decision Tree*. We trained our model on 80% of the data points (randomly selected), holding 20% in reserve for accuracy testing. To ensure robustness of the resulting predictive accuracy, we performed a 5-way cross validation.

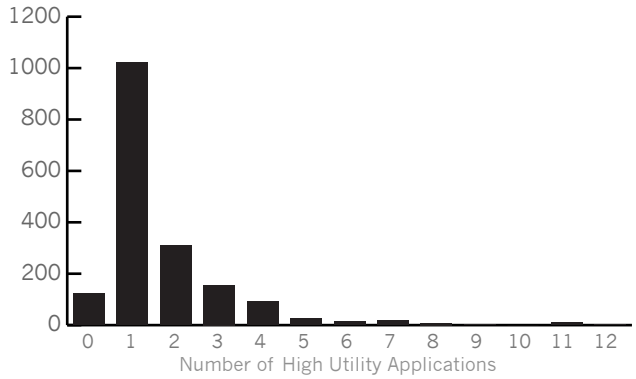

Figure 5. Histogram of Number of High Utility Applications
note similarity to Pareto distribution

To strengthen our understanding and test the accuracy of the models created, we compared them to two other prediction approaches: *Approach 1-* a naive guessing strategy (*Always Guess 0*) which, due to the high percentage of low utility applications, would always choose low utility; and *Approach 2-* associating *high-utility applications* with OS application focus. This approach is often used in prior work, and serves as another comparison for our models.

Comparisons between each predictive model's perceived application utility were calculated using Cohen's kappa statistic and 95% confidence intervals, following standard machine learning techniques [34]. Cohen's kappa calculates the agreement between the user's marked application utility and the model's prediction. This is more accurate than percentage agreement, in that it takes agreement occurring by chance into account. Confidence intervals describe the probability that a correct value will be produced. A 95% confidence interval is a p-value of 0.05.

RESULTS

The mean number of *high-utility applications* across all users was 1.69. When examined as a set of tabulated frequencies (Figure 5), it is apparent that 35.62% of the time, users had more than one *high utility application*. Thus, if a binary feature could perfectly divine application utility (e.g., which application has focus), it will fail to predict all of the *high-utility applications* 35.62% of the time, because only one application can have focus at any given moment. Given this context, we examined our models' accuracy.

Strength of Models

The technical definition of a model's Accuracy score is percent agree / total. Our *Decision Tree* has the highest value (89.13%) followed by *Application Focus* (87.06%) and *Logistic Regression* (86.76%). Naive Bayes also has a high score (82.66%), almost equal to *Always Guess 0,* which blindly ranks all applications as low utility (81.87%). Although each of these models have a high Accuracy score, *Always Guessing 0* (which is useless in practice) is equally high. This is a direct result of the large percentage of low utility applications. **The Accuracy scores are greatly inflated, rendering them useless as a measure of each model's ability to predict high-utility applications — the most relevant feature to the user.** This is a known issue with using accuracy as a metric of model quality in data mining and information retrieval [4].

	Kappa	Conf. Interval 95%	Conf. Interval 99%
Always Guess 0	0.00	[na]	[na]
Utility is App Focus	0.48	[0.44, 0.52]	[0.42, 0.53]
Naive Bayes	0.39	[0.35, 0.43]	[0.34, 0.45]
Logistic Regression	0.53	[0.49, 0.57]	[0.48, 0.58]
Decision Tree	0.62	[0.58, 0.65]	[0.57, 0.67]

Table 1. Accuracy of Predictive Models
Kappa scores, 95% and 99% confidence intervals

Consider a Monty Hall-type problem with 100 doors and only one prize. If you guess all doors to have no prize, you will have a 99% accuracy for guessing what a door contains. This results in a high Accuracy score (only 1% away from 100% accuracy), yet you will *always* lose the game (since you would never take a chance on selecting one of the doors). In other words, 99% accuracy is not always "good," depending on the context. Similarly, if we are identifying *high-utility applications*, having a 82% accuracy by always guessing 0 (low-utility) will have a high Accuracy score, yet will be an ineffective model since the real strength of these predictive models lie in correctly identifying the *high-utility applications*. Thus, the Accuracy score likely obscures the true agreement between the predictive and perceived application utility of the model.

Cohen's kappa is a statistical measure of accuracy (giving weight to positives *and* negatives), and presenting a measure more in tune with the behavior of a model against an accepted standard. Kappa also takes into account events agreeing by chance (whereas Accuracy does not). Table 1 presents kappa scores and confidence intervals for each model. Using the kappa statistic, stark differences between the models' predictive ability are seen. *Decision Tree*, the most complex model, has a kappa score of 0.62, considered "Substantial" or "Good" depending on the classification scheme used[32] as compared to using Application Focus considered "Fair" with a kappa of 0.48. Moreover improvements between *Decision Tree*, and all other models (including *Window Focus*) are all highly significant ($p<0.01$). In other words, *Decision Tree* has the highest agreement with the original data. This improvement is statistically significant compared to Naive Bayes, Logistic Regression, Always Guess 0 (low utility), and Utility is Application Focus.

Perhaps the most representative measure of the model's ability to predict *high-utility applications* is through a sensitivity analysis (Table 2) [4]. Sensitivity is the proportion of *high-utility applications* that were correctly identified as such. *Decision Tree* had a sensitivity of 66%. This improves on the sensitivity of *Application Focus* by 53%. In other words *Decision Tree* is significantly more sensitive than the current approach, using *Application Focus*. However, sensitivity comes with a slight cost of lower specificity (more false positives). This finding is similarly supported in Table 2, through a recall precision test[37], commonly used in analyzing web and document retrieval accuracy. *Decision Tree* was shown to have a good recall score

	Sensitivity	Specificity	Recall	Precision
Always Guess 0	0.00	1.00	0.00	0.00
Utility is App Focus	0.43	0.97	0.75	0.43
Naive Bayes	0.47	0.90	0.52	0.47
Logistic Regression	0.58	0.93	0.65	0.58
Decision Tree	0.66	0.95	0.72	0.66

Table 2. Additional Model Analysis
all values are percentages

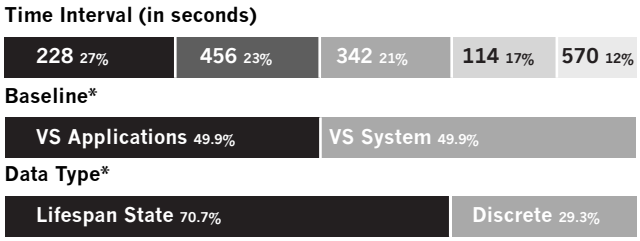

Figure 6. Strength of Predictive Measure Dimensions
** denotes some categories could not be viewed, <0.01% weight*

finding the events that matter) but a lower precision score (due to more false positives).

Strength of Predictive Measures

Performing a logistic regression provides an insight into the weighting of the 11,899 predictive variables. The weight of each variable can be computed by summing the absolute values of its coefficients. While not an exact replication of the weighting and model in the regression (or in the more accurate decision tree), extracting the more salient variables can better guide future model design and system construction. We find that 50% of the predictive power of the logistic regression can be derived from the top 22 variables, with 75% power from the top 47, 90% from the top 95, and 99% from the top 238 predictive variables. Figure 6 illustrates the predictive power breakdown of each dimension. Because each predictive variable has one baseline, one time interval, and one data type, the power breakdown for each dimension includes all the variables. Most notable in Figure 6 is the strong power of *Lifespan State category*. When further expanded, 4 main sub-categories emerge as having large predictive weight. Table 3 highlights those top predictive *Data* Type subcategories, as well as the overall top 5 predictive variables.

Top 5 Data Type Subcategories	Weight %
Total/Longest Time Mouse was Over an Application in Focus	25.61
Longest Consecutive Time Application is Running with at Least One Window in Another Space, at at Least One Window Currently on Screen	21.00
Total Time Application is Running with at Least One Window in Another Space, at at Least One Window Currently on Screen	21.00
Lifespan Variables Time Application is in Focus	19.56
Lifespan Variables when Application is Running	3.17

Top 5 Predictive Variables (Overall)	Weight %
Total Time Application is Running in another space with at Least One Window Currently on Screen over the past 456 Seconds	3.21
Total Time Application is Running in another space with at Least One Window Currently on Screen Versus any Application over the past 456 Seconds	3.21
Longest Consecutive Time Application is Running in another space with at Least One Window Currently on Screen over the past 456 Seconds	3.21
Longest Consecutive Time Application is Running in another space with at Least One Window Currently on Screen Versus any Application over the past 456 Seconds	3.21
Total Time Application is Running in another space with at Least One Window Currently on Screen over the past 342 Seconds	3.21

Table 3. Predictive Power of Top Variables

DISCUSSION

Our results show that with a relatively low-cost *Decision Tree* model, we can build an accurate application utility function (66% of actual user-specified *high-utility applications* are predicted as being high-utility by the model). More importantly, this is a 53% increase over the current method for predicting high application utility ($p<0.01$). Based on the sensitivity of the using *application focus*, the existing approach as a 57% failure rate. In other words, the accuracy of the generated predictive models demonstrates a strong potential for computerized systems to accurately predict *high-utility applications*. With such an increase in predictiveness, the ability of other fields of HCI and Computer Science to accurately predict Interruption time, Task, Workflow, and Time Management is also increased.

It should be noted that though the notion of *high-utility applications* is subjective, we were able to build a quantitative model that accurately predicts *high-utility applications*. Moreover, given the wide background of our participants (education, age, domain, etc) it appears that there may be some degree of a universal definition of *high utility*, that can be quantitatively assessed, measured, and calculated.

When considering the models themselves, it becomes evident that not all variables may be necessary to create an accurate utility function. It is interesting to note that the tree model utilized a small subset (162) of the 11,899 variable corpus. With the tree model using ~1% of the variables and a logistic regression using between 0.1% and 2% (depending on power desired), we can quantitatively show the relatively small set of truly predictive variables.

The majority of high power variables have a low computational overhead. These predictive variables revolve around application and window lifetime, and the number of windows (all of which are simple to calculate). On the other hand, the run time of calculating more complex metrics, such as determining the visible window area of an application, can grow exponentially as the number of applications and windows increase. While this model has not been tested in real time, we hypothesize that these metrics can be estimated by randomly sampling a small portion of the pixels on screen rather than calculating the exact area. While this will not be as accurate, the computational overhead will be minimal at worst. Therefore, computer systems can use a far more accurate model with little or no cost to system overhead beyond using simple window focus.

An interesting observation of many of the top predictive variables to help identify *high-utility applications* is the position of the mouse over windows (e.g., total time mouse was over application while application was in focus; longest time mouse was over window while window was visible). This is not unexpected, in that the mouse is used to interact with relevant application windows, we can therefore infer that the mouse being present on a window indicates the user's intention to act soon, or their recent application interactions.

Another feature that was frequent in the top 50 variables involve a user's windows being present in multiple simulated desktops (or spaces). We do not have qualitative feedback from users as to the significance of this. However, we can provide some informed speculation. Much like in Linux, multiple spaces are designed in OS X to allow users to multi-task, and group their application windows according to a theme or activity. If we consider applications that are present in more than one space (activity), that application may be playing a significant role in achieving multiple aims.

These results, however, do present an unanswered question: how much error can a user tolerate? As prediction accuracy increases for *high-utility applications*, false positive rates also appear to increase as well. Much like in web-search, there is a threshold of error at which the "clutter" of false positives drowns out the significant findings. Where that line lies is an unexplored, but very important question. By understanding and quantifying the impact of false positives (and false negatives), we can build a cost function into future models, penalizing the system for different types of errors, thus improving accuracy, precision and recall.

Implications

Accurately determining application utility is a critical first step to contextually sensitive systems in Task Analysis, Intelligent Support Tools, and Interruption Research. Given the 57% failure rate of existing approaches which rely upon unsubstantiated metrics, our solution provides an accurate alternative based on concrete real-world data analysis. Our work improves 53% in the prediction of *high-utility applications*, providing researchers and software developers a more robust framework to build tools for task analysis, context aware applications, and interruption tools.

Consider aware computing, which attempts to provide users with meaningful support during complex activities. If the software can determine which applications are in use, systems can better determine from where to extract text for automated resource queries, providing more salient information to the user. Similarly, time management software will be able to better calculate applications that are the recipients of the users attention, rather than relying on a system's binary focus. Lastly, a large corpus of task analysis research has attempted to provide users with automated groupings of running applications. By integrating these predictive models into their systems, the returned collection of "related" applications will more accurately reflect the applications that are of high utility to the user.

LIMITATIONS

From the outset, our goal was to create a general utility function from as wide a sample of the population as possible (ages, gender, educational attainment and profession). While we achieved this goal, this work is based in the Apple OS X environment, which has its own nuances, as does every operating system. We look forward to evaluating our approach on other platforms to compare these findings.

FUTURE WORK

While the current investigation with CLOTHO was quite revealing, there are many avenues for future exploration, both using the existing data set, as well as opportunities to apply similar techniques to answer questions such as assigning utility to individual windows, or tabs within a multi-tab application (e.g., web browser). Further, can complex machine learning algorithms be applied to this data set (e.g., SVM) to improve accuracy, or is there an upper bound on prediction system performance?

A corollary to this study would be to allow for debriefing in situ. The focus of this paper was to examine the perceived application importance in recall. A similar study could build a model of application utility in the moment, and compare the weightings of both models. This could provide quantitative support to Norman's article in Interactions [25].

With such a rich corpus of data collected from the 22 users, many questions can be investigated to further uncover aspects of perceived application utility and measure human computer interactions. For example, how does distance (temporally) from the event change the model of application utility? In other words, this paper examined data prior to the debriefed event; could data that occurs after the event in question be used to (retroactively) more accurately predict utility? Further, what is the impact of demographic on predictive ability and model?; academia vs. industry or male vs. female?

With web applications becoming increasingly prevalent, as well as the ability to change "applications" within one window, our perception model for web browser applications may change. Unlike other groups of applications, the web browser set is finite for all practical purposes (*Safari*, *Chrome*, *Firefox*, *Camino, Opera*, and *Internet Explorer)*. Given this categorization and unique styles of interactions, performing similar analysis could yield interesting findings. Further, examining application utility from the perspective of an individual window is also necessary as users can have multiple text editor windows open at a time, or web browsers can point to a variety of web-applications each with a different utility value to the user.

In addition, there are non-computer data sets that may be worth incorporating into a predictive model such as phone use, locational data (home vs. office), and even the presence of other people in the work space. Consideration of variables such as lighting, weather, and peripherals may yield information as to the use and utility of applications.

CONCLUSION

Research in multiple domains of HCI rely upon an a-priori assessment of high application utility to accurately classify tasks, identify break points, and analyze user workflows. Given the 57% error rate in *high utility application* detection, these high-level analysis are subject to increased error just as a malfunctioning thermometer will degrade the quality of a scientific experiment. In this paper, we have examined the interrelationship between generalized system resource consumption, and perceived application utility by users in order to create a more accurate metric for determining *high-utility applications*.

We utilized a large real-world data set (22 users and 321 real-world hours of data) and constructed a set of models for predicting application utility with direct applications to HCI in Task Analysis, Interruption Research and Workflow Analysis. We demonstrate a clear improvement of 53% over existing techniques ($p<0.01$) with high sensitivity (66%) and kappa scores (0.62). By examining a data set from a wide age, background, and occupation of users, we believe our findings to have broad applications across multiple domains. While there may be benefits from adding in application specific features to our model, the strength of our system shows that even with application independence, we can create a robust model of application utility.

We believe this work addresses a fundamental challenge for system designers that wish to provide context sensitive systems based upon the applications currently in use. While this research is not definitive, and allows for many avenues for future investigation, we believe this work takes important first steps towards providing a more comprehensive understanding of user behavior and interaction with computer systems by demonstrating a significant 53% improvement over existing practice.

ACKNOWLEDGMENTS

We would like to thank Joseph Scubida, Nicholas Jitkoff and Wai-Tat Fu, for all their advice and guidance, our participants without whom this research would not be possible, and our friends and family for putting up with our unforgivable work schedules.

REFERENCES

1. Adar, E., Teevan, J., and Dumais, S. T. Large scale analysis of web revisitation patterns. In Proc. SIGCHI CHI 2008 Conference (2008). ACM, New York, NY.
2. Bruegge, B. et al. Classification of tasks using machine learning. In Proc. PROMISE '09 Conference (2009). ACM, New York, NY.
3. Brush, A. J. B. et al. Understanding memory triggers for task tracking. In Proc. CHI '07 Conference (2007). ACM, New York, NY.
4. Manning, C. D., Raghavan, P., and Schuetze, H. Introduction to Information Retrieval. Cambridge University Press, Cambridge, England, 2009.
5. Consolvo, S. and Walker, M. Using the Experience Sampling Method to Evaluate Ubicomp Applications. IEEE Pervasive Computing, 2(2), 2003, 24–31.
6. Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. In Proc CHI '04 Conference (2004). ACM, New York, NY.
7. Diaper, D. and Stanton, N. The handbook of task analysis for human-computer interaction. CRC, 2003.

8. Diaper, D. and Stanton, N. A. Review of "The Handbook of Task Analysis for Human-Computer Interaction". INTERACTIONS, 13(3), 2006, 62–63.
9. Dragunov, A. N. et al. TaskTracer: a desktop environment to support multi-tasking knowledge workers. In Proc. IUI '05 Conference (2005). ACM, New York, NY.
10. Fenstermacher, K. D. and Ginsburg, M. A Lightweight Framework for Cross-Application User Monitoring. Computer, 35(3), 2002, 51–59.
11. Grudin, J. Why CSCW applications fail: problems in the design and evaluationof organizational interfaces. In Proc. CSCW '88 Conference (1988). ACM, New York, NY.
12. Gyllstrom, K., Soules, C., and Veitch, A. Activity put in context: identifying implicit task context within the user's document interaction. In Proc. IIiX '08 Conference (2008). ACM, New York, NY.
13. Hilbert, D. M. and Redmiles, D. F. Extracting usability information from user interface events. ACM Computing Surveys, 32(4), 1999, 384–421.
14. Iqbal, S. T. and Bailey, B. P. Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. In Proc. CHI '07 Conference (2007). ACM, New York, NY.
15. Jin, J. and Dabbish, L. A. Self-interruption on the computer: a typology of discretionary task interleaving. In Proc CHI '09 Conference (2009). ACM, New York, NY.
16. Kaptelinin, V. UMEA: translating interaction histories into project contexts. In Proc. CHI '03 Conference (2003). ACM, New York, NY.
17. Karger, D. R. et al. Haystack: A customizable general-purpose information management tool for end users of semistructured data. In Proc. 2005 CIDR Conference (2005). Asilomar, CA.
18. Kellar, M. and Watters, C. Using web browser interactions to predict task. In Proc. WWW '06 Conference (2006). ACM, New York, NY.
19. Kelly, D. and Belkin, N. J. Display time as implicit feedback: understanding task effects. In Proc. SIGIR '04 Conference (2004). ACM, New York, NY.
20. Kirwan, B. and Ainsworth, L. K. A guide to task analysis. Taylor & Francis, 1992.
21. Lee, U., Liu, Z., and Cho, J. Automatic identification of user goals in Web search. In Proc. WWW '05 Conference (2005). ACM, New York, NY.
22. Lim, Y.-K. Multiple aspect based task analysis (MABTA) for user requirements gathering in highly-contextualized interactive system design. In Proc. TAMODIA '04 Conference (2004). ACM, New York, NY.
23. ManicTime.com ManicTime. 2008,
24. Matthews, T. et al. Clipping lists and change borders: improving multitasking efficiency with peripheral information design. In Proc CHI '06 Conference (2006). ACM, New York, NY.
25. Norman, D. A. THE WAY I SEE IT Memory is more important than actuality. interactions, 16(2), 2009, 24–26.
26. Oliver, N. et al. SWISH: semantic analysis of window titles and switching history. In Proc. IUI '06 Conference (2006). ACM, New York, NY.
27. Pinelle, D., Gutwin, C., and Greenberg, S. Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. ACM Trans. Computer-Human Interaction, 10(4), 2003, 281–311.
28. Rattenbury, T., Nafus, D., and Anderson, K. Plastic: a metaphor for integrated technologies. In Proc. UbiComp '08 Conference (2008). ACM, Seoul, Korea.
29. Robertson, G. et al. Scalable Fabric: flexible task management. In Proc. AVI '04 Conference (2004). ACM Press, New York, NY.
30. SLife Labs. SLife. 2009, http://www.slifelabs.com/.
31. Stary, C. and van, d. V., Gerrit C. Task analysis meets prototyping: seeking seamless UI-development. In Proc. CHI '99 Conference (1999). ACM, New York, NY.
32. Szklo, M., Nieto, F. J., and Miller, D. Epidemiology: beyond the basics. In American Journal of Epidemiology (2001). Oxford Univ Press,
33. Tak, S. et al. Improving Window Switching Interfaces. In Proc. INTERACT 2009 Conference (2009). New York, NY.
34. Tan, P. N., Steinbach, M., and Kumar, V. Introduction to data mining. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, 2005.
35. Van, D. V., G.C., Lenting, B. F., and Bergevoet, B. A. J. GTA: Groupware task analysis—Modeling complexity. Acta Psychologica, 91(3), 1996, 297–322.
36. Voida, S., Mynatt, E. D., and Edwards, W. K. Reframing the desktop interface around the activities of knowledge work. In Proc UIST '08 Conference (2008). ACM, New York, NY.
37. Witten, I. H. and Frank, E. Data mining: practical machine learning tools and techniques with Java implementations. ACM SIGMOD Record, 31(1), 2002, 76–77.