# Identifying Categories of End Users Based on the Abstractions That They Create

Christopher Scaffidi, Andrew Ko, Brad Myers, Mary Shaw
December 2005
CMU-ISRI-05-110
CMU-HCII-05-101

Institute for Software Research, International, Carnegie Mellon University
Human-Computer Interaction Institute, Carnegie Mellon University

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

**Keywords:** end user programming, end user software engineering, end users, adoption, abstraction

## Abstract

*Software created by end users often lacks key quality attributes that professional programmers try to ensure through the use of abstraction. Yet to date, large-scale studies of end users have not examined end user software usage at a level which is sufficiently fine-grained to determine the extent to which they create abstractions.*

*To address this, we deployed an online survey to Information Week subscribers to ask about not only software usage but also feature usage related to abstraction creation. Most respondents did create abstractions. Moreover, through factor analysis, we found that features fell into three clusters–when users had a propensity to use one feature, then they also had a propensity to use other features in the same cluster. These clusters corresponded to macro features, linked data structure features, and imperative features.*

*For each of the three factors, we created a scale and used it to categorize users into two bins—those with a high propensity to use features associated with that scale, and those with a low propensity. Compared to users with a low propensity to use imperative features, users with a high propensity to use imperative features were more likely to report testing and documenting. Propensity to use linked structure features was less strongly related to these practices. These findings represent a step toward a more complete map of end users' skills.*

This work has been funded in part by the EUSES Consortium via the National Science Foundation (ITR-0325273), by the National Science Foundation under Grant CCF-0438929, by the Sloan Software Industry Center at Carnegie Mellon, and by the High Dependability Computing Program from NASA Ames cooperative agreement NCC-2-1298. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

# 1. Introduction

Producing software with high reliability, maintainability, composability, and other quality attributes is a challenge for both professional programmers and end user programmers. To help ensure these quality attributes, professional programmers have developed a number of techniques, chief among them creating, testing, and documenting abstractions [3] [18].

Since these practices have proven invaluable to professional programmers, it is natural to ask whether *end user programmers* engage in similar practices. Therefore, we surveyed readers of *Information Week* to answer two questions:

- To what extent do end user programmers create abstractions?
- Are end user programmers who create abstractions more likely to test and document their work, compared to end user programmers who do not create abstractions?

To address these two high-level questions, we asked respondents about usage of specific application features related to creation of abstractions within a wide range of end user programming environments: web browsers, web server scripting engines, databases, spreadsheets, and other desktop applications. We asked about 23 abstraction-related features, including JavaScript functions, reusable server-side "include" files, database tables, and macros. For example, to learn whether end users create reusable computational abstractions in a browser environment, we asked whether respondents whether they created JavaScript functions.

With respect to the first bulleted question above, we found that not all abstraction-related features are heavily used, even by this computer-savvy population of *Information Week* readers. For example, only 16% of respondents reported using server-side "include" files (which are useful for defining headers, footers, and other chunks of HTML that are the same for all pages in a web site). But near the other end of the spectrum, 89% of respondents reported using functions (like "sum") to link cells of spreadsheets.

To characterize the structure of feature usage by end user programmers, we applied factor analysis, which revealed that features fell into three clusters. For each cluster, people with a propensity to use one feature in that cluster tended to have a higher propensity to use other features in the same cluster. The three clusters were propensity to create macros, propensity to design linked structures, and propensity to write imperative code. For example, people with a propensity for using one imperative feature (such as JavaScript functions) also had a propensity for using other imperative features (such as stored procedures).

In addition, as noted in the second bullet above, we wanted to learn whether end users who created abstractions were more likely to test and document their work, compared to end user programmers who do not create abstractions. In preparation for answering this question, we constructed a scale for each of the three clusters above to measure relative propensity to use features within that cluster. We then classified each respondent as "above average" or "below average" on each scale.

ANOVA tests revealed that people with an above average propensity to create imperative abstractions were more likely to test and document their work (compared to people with a below average propensity to create imperative abstractions). In addition, they reported more knowledge of programming terms like "loop" ($P \leq 0.02$ for all these results).

When we divided respondents into above average and below average groups based on their propensity to create linked structures, the differences between above-average and below-average groups were not as stark as the differences that we saw when we divided users based on usage of imperative abstractions. Specifically, people with an above average propensity to create linked structures were more likely to document their work but only marginally more likely to test their work (compared to people with a below average propensity to create linked structures). They also reported more knowledge of programming terms ($P \leq 0.03$ for all these results).

Finally, when we divided respondents into above average and below average groups based on their propensity to create macros, we saw no statistically significant differences between these two groups, in terms of their propensity to test or document their work (at $P = 0.05$). There was no statistically significant difference in their knowledge of programming terms, either (at $P = 0.05$).

In short, these analyses have revealed substantial structure within the feature-usage patterns of end user programmers. Further studies will be required in order to evaluate possible causes for this structure and to assess how creating, testing, and documenting abstractions affects software quality attributes in the case of end user programming.

## 2. Related Work

There have been many studies of abstraction use by end users, but the majority of them have addressed abstraction usage and adoption in a single type of application or domain in the short time span of a user study. For example, several of the abstractions in the Forms/3 spreadsheet language have been studied from an adoption perspective, to assess what factors might influence adoption and continued use of the new abstractions [26][27]. This has led to a design strategy called "surprise-explain-reward" in which systems attempt to (1) surprise users by identifying some knowledge gap, (2) explain the knowledge gap to users, and finally (3) reward users when they use a new abstraction successfully. There have been several other attempts to design abstractions for particular populations of end users or particular types of data. The HANDS environment was designed for programming by children [16] by studying patterns in the language that children used to describe solutions to programming problems. This led to a programming language that children could quickly learn and use to create interactive simulations. The abstractions in the Lapis environment were designed specifically for manipulating textual information [13], simplifying many common end user tasks such as search and replace. The Pursuit environment [14] offered graphical abstractions for writing file management scripts. Although systems have been built for a variety of tasks and data, there have been few studies of the systems' long-term usage or the use of other abstractions in tandem.

Many end-user programming systems have been informed by Blackwell's "Attention Investment" theory [2], which argues that feature adoption is a factor of users' perceived cost and reward of an unfamiliar abstraction. Of course, many factors can influence these perceptions, including gender and self-efficacy [1], co-workers' knowledge of software applications, and the degree to which abstractions can support each other between applications. Many of these factors were detailed in Nardi's ethnographic work investigating the use of spreadsheets and other tools in various workplaces [15]. For example, she identified one type end user programmer that she calls "tinkerers," which often play the role of introducing less experienced co-workers to unfamiliar abstractions. Because Nardi used semi-structured interviews, the range and prevalence of the trends that she observed are unknown.

Studies of people using abstractions outside of lab contexts have largely focused on a single domain of end-user programming at a time. For example, Rode et. al. have conducted interviews with end users in the home to learn what abstractions are supported by home appliances [20], finding two types of widely used categories of abstractions, named "ahead of time" and "repeats easy." Fisher and Rothermel [4] have studied public Excel spreadsheets available on the web in order to assess the prevalence of macro and formula use, finding that very few spreadsheets use formulas, and if they do, they use relatively simple mathematical operators and functions. While the corpus is quite helpful for analyzing the usage of specific abstractions in spreadsheets, the corpus has no data about the individuals responsible for the spreadsheets, or what other abstractions they may also use as part of their work activity. Rosson et. al. have focused on the use of web-programming environments by end users, investigating the motivations and goals behind their use of HTML and other languages [21]. They found that many web programmers became web programmers out of circumstance rather than explicit choice (for example, taking over the management of a web site for a friend). Although they were able to collect more personal information in their surveys, they did not study their respondents' use of particular abstractions in web applications, such as links and server-side includes.

To our knowledge, the only other wide-scale studies that inform this work are secondary analyses of US Bureau of Labor Statistics studies of application use [25], which indicate that over 45 million people used spreadsheets and/or databases at American workplaces in 2001, and this number is expected to climb to 55 million by 2012. The aim of the survey we present in this paper is to begin to characterize and cluster these individuals as well as other end user programmers, providing additional details about the features and abstractions that they use.

# 3. Survey

This section discusses the questions that we presented to *Information Week* readers, and it discusses the demographic characteristics of the respondents, particularly the fact that many work in IT departments and have substantial programming skills. This section also notes a low response rate (under 1%) and some small non-response skew in demographic characteristics. All these factors may limit the generalizability of our results. Thus, we anticipate needing to deploy follow-up studies, as outlined in Section 6.

## 3.1 Motivation and Context

Existing characterizations of end user populations based on software usage provide only minimal guidance on how to help end user programmers practice better software engineering. A coarse-grained categorization based on software usage is inadequate for guiding programming tool designers: it tells *what* tools people use but not *what portions* of those tools receive heavy use. It glosses over communities of end users with special needs or capabilities, and it fails to highlight concerns spanning multiple types of environments.

Hence, in this survey, we wanted to go deeper than software usage and also ask about feature usage. In particular, we have chosen to ask about usage of features that might be reasonably construed to relate to programming. While definitions of "programming" vary (see [23] for a summary), features of interest certainly include those related to creation of JavaScript functions, hyperlinks, stored procedures, database tables, and spreadsheet macros.

Moreover, we chose to focus on programming features related to the representation of abstractions. We were led to this focus by a realization that abstraction is one of the principal techniques that professional programmers have developed in order to support the creation of high quality code. For example, high notation viscosity (the difficulty of making local changes) can damage software maintainability, but it is known that "viscosity can be reduced by increasing the number of abstractions" [6] This maintainability-enhancing design principle first appeared in the software engineering literature over thirty years ago [19]; likewise, researchers realized long ago that comprehensive testing requires modular code [3] [18]. However, abstractions had also been identified as difficult to learn [5] [17], so many end user programming environments seek to reduce the need for end users to be consciously aware of using abstractions [10] [16].

When designing the study, the three main abstractions we had sought to study were variables, functions, and data structures. Researchers have provided support for these abstraction representations in a variety of end user programming environments (as we have discussed in [24]), so there seems to be an instinctive recognition among researchers that these are important features. We doubt that end users are likely to make heavy use of more advanced abstraction representations like abstract data types and inheritance hierarchies of classes, and we have seen nothing in the results of this survey to make us suspect otherwise. (Although we saw feature-usage clusters related to functions and data structures as anticipated, we did not find a variable-specific cluster. Instead, we found a macro-related cluster. These findings are discussed in more detail by Section 4.)

In order to evaluate how abstraction usage spans multiple environments, we have asked about comparable features across a variety of software environments, hoping to detect clusters of logically related features that end users have a propensity to use. As described in Section 4, we did indeed observe such clusters suggesting that certain feature usage propensities do co-occur. This finding could be caused by a variety of underlying phenomena ranging from users' understanding to business processes. Thus, we anticipate needing to follow these surveys with interviews, tests, or other studies to better understand why clusters exist.

To conclude, our survey represents an enhanced method of characterizing the end user population, based on categorizing end users according to the ways they represent abstractions. Since the use of abstraction can facilitate key software engineering goals (such as improving reusability and maintainability), this categorization promises an improved ability to highlight niches of end users with special software engineering capabilities or struggles.

## 3.2 Survey Instrument

Our survey contained four parts, the first of which asked users whether they or their subordinates had used each of six pieces of software in the past three months. The main five classes of interest were databases, web page editors, web server scripting languages, spreadsheets, and word processors / slide presentation editors. (The sixth class, business intelligence software, was used by only 22% of respondents and will not

be discussed further in this report.) If the respondent indicated usage of a piece of software, then we asked about feature usage within that software during the past three months. These feature questions were designed to help evaluate whether the respondent or subordinates were engaging in programming-like activities. For example, if the respondent indicated usage of web page editing software, then we asked if the respondent or any subordinates created JavaScript functions.

In the second part of our survey, we asked about the relationship between the respondent's work and software usage. In this report, we focus on six questions wherein we asked the respondent to consider how useful the web was during the previous year. Specifically, we asked about the usefulness of the web as a source of examples, documentation, and numerical data, and also as a destination for publishing examples, documentation, and numerical data.

The third part of our survey asked about familiarity with certain programming terms: variables, subroutines, conditionals, and loops. For each programming term, we provided a brief definition and, if appropriate, a short list of synonyms that the user might be more familiar with. If the respondent indicated familiarity with the programming term, then we asked if he or she had personally used the corresponding construct in the past year. For the "subroutines" term, we asked two usage questions rather than just one, in order to distinguish between creation of subroutines and usage of subroutines created by other people.

Finally, the fourth part asked a number of questions concerning the respondent's firm and individual background, with a particular emphasis on education and training. In retrospect, we would have liked to include a number of additional questions, particularly gender and age, which we will include in any future surveys (see Section 6).[1]

### 3.3 Survey Delivery

In late 2004 and early 2005, we partnered with *Information Week* magazine to develop our online survey of technology-savvy individuals at American firms. As discussed in Section 3.2, the questions of this survey primarily focused on usage of software features by respondents and their subordinates. In this section, we describe the process by which we designed and deployed the survey, and how we cleaned the data.

The over-arching consideration in designing the survey was our desire to eventually discover whether abstraction might help end users produce better software—and if so, then what kinds of abstraction might show the most promise. One key precondition for this, however, is to understand whether end users can use abstractions at all. Hence, the primary goal of the current survey is to begin documenting the extent and variety of end user abstraction within existing popular tools.

The other consideration in designing our survey was to evaluate questions and analysis techniques that might prove useful in a later survey of a targeted sample. Consequently, in designing the current survey, we chose questions that covered the "lowest common denominator" for a wide variety of workers. We also included some questions, such as those dealing with business intelligence systems, that *Information Week* wished to deploy for their own purposes.[2]

We tested the survey by presenting it to three marketing professionals. We delivered the questions to two of these people via a simple web implementation that also provided a "scratch pad" where they could enter comments concerning the questions; we presented the questions to the third person verbally, which allowed us to perceive facial expressions and discuss the questions if they were unclear. These practice runs raised a number of issues that provoked some question rewording. For the most part, we simply needed to add clarifying parenthetical comments; for instance, in the question about whether respondents used spreadsheet functions when typing in cells, we added a parenthetical comment explaining that "sum" exemplified the sort of function we were asking about. Finally, *Information Week* implemented the survey using the Inquisite web-based online survey system [8] and tested it several more times using members of their staff, which provoked a few final layout changes.

---

[1] For the text of our 96 questions, refer to Appendix A.

[2] *Information Week*'s report, which is based on the subset of 378 respondents who *Information Week* classified as business technology professionals, is available at
http://www.informationweek.com/story/showArticle.jhtml?articleID=163102092

*Information Week* announced the survey by sending 125,000 emails to subscribers who had previously indicated their interest in receiving such announcements. As an incentive for participating, respondents were entered into a drawing for one first prize of $500 and five second prizes of $100 each. Within two months, 831 people completed the survey.

*Information Week* also published a link to the survey on their web site. Although they did not instrument the survey URL to record whether respondents found the survey via the emails or via the main web site, virtually all respondents took the survey soon after the emails went out. (Over 95% took the survey within one week of the emails, even though the survey was deployed for approximately two months.) Consequently, it is safe to assume that the sample frame generally corresponded to the *Information Week* database from which the email addresses were selected. To assess non-response bias, we compare our sample to that database in Section 3.4.

We received the data from *Information Week* in comma-separated-value format and imported it into SAS 9.1.3. Skipped questions were flagged as missing, as were "I Don't Know" answers. Since the survey was deployed via the web, and since most of our questions were multiple choice, very few items contained unintelligible entries.

The main exception was the question about how many subordinates the respondents had. Due to the implementation of the survey, respondents were able to enter arbitrary text into this item (such as "varies"). Moreover, some answers were ambiguous as to how many subordinates the respondent had (if any). Unfortunately, we also found that respondents with a large number of subordinates tended to report extremely high levels of feature usage in essentially every category; consequently, after viewing various plots of feature usage versus subordinate count, we decided to discard data from respondents with 1000 or more subordinates, as well as any respondents for which we could not interpret the subordinate count. Of our original 831 respondents, this left 816.

We developed a coding system for the handful of demographic open response variables used in Section 5.2, such as college major and industry. In general, these were relatively unambiguous to code, so we did not implement any inter-rater checks; moreover, none of these variables showed any hint of being related to any of our statistically significant results. Consequently, although this survey has provided good guidance for converting these questions into multiple choice in any future surveys, we may consider omitting these questions altogether.[3]

Finally, during our data cleaning process, we took the opportunity to perform cross-checks between related items. For example, we included two copies of a question about creation of web forms (once in the context of web page editors, and once in the context of web server scripting). If the user answered "No" to one of these items but "Don't Know" or skipped the other, then we set both answers to "No" (and analogously for the situation involving "Yes" and "Don't Know"). Lastly, if the user answered "Yes" to one but "No" to the other, then we changed both to "Don't Know." These modifications affected less than a dozen respondents.

## 3.4 Sample Characteristics

In general, the people who responded to our survey reported a great deal of technical knowledge. This was not surprising, since the sample frame of respondents in the database that *Information Week* cultivates also tended to be technically savvy, as many of them worked in Information Technology (IT) departments. Due to non-response, our respondents tended to be slightly more likely to be consultants and somewhat more likely to work for small firms. Yet, as described in Section 5.2, our results do not appear to be strongly dependent on these demographic variables, so this survey's results may generalize to the entire sample frame. Of course, in order to generalize to a larger section of the American workforce, the results of this survey would require confirmation through a follow-up survey with a more controlled sample, as discussed in Section 6

[3] We did, however, implement a more robust coding process for certain open-response variables, particularly those related to usage of help systems by respondents. These items are not central to the present report and may be discussed at a later date.

### 3.4.1 Characterization of Respondents

We found that respondents generally reported significant technical background. Most respondents possessed an educational background that would be expected to give them strong technical skills. All respondents had at least a high school education, and 79% had at least a Bachelor's degree. Moreover, 26% of all respondents had a computer-centric major such as computer science, information technology, or e-commerce; an additional 40% of all respondents had a math-heavy major, such as mathematics, engineering, science, or accounting.

Given this educational background, it is not surprising that respondents generally were familiar with several programming concepts: variables, subroutines, conditionals, and loops. In fact, 79% of respondents were familiar with all four terms. Moreover, respondents generally reported actually using the corresponding programming constructs. In the past year, approximately 35% of all respondents created variables, used co-workers' subroutines, created new subroutines, coded conditionals, and coded loops. With such high programming knowledge among respondents, it is not surprising that most features related to abstraction-creation were used by many of the respondents, as shown in Table 1.

**Table 1: We found that most (but not all) features were used by many respondents (N=816). The top ten feature rates are bolded below.**

| Software Category | Software Usage (% of all users) | Feature Variable | Theme of Feature Variable | Feature Usage (% of all users) |
|---|---|---|---|---|
| Server-Side Code | 52.3 | svr_include1 | Using Web server-side static includes | 16.3 |
| | | svr_include2 | Referencing PHP/Perl libraries | 27.9 |
| | | svr_new | Using PHP/Perl "new" function | 25.4 |
| | | svr_func | Creating web server script functions | 40.9 |
| Web Pages | 68.6 | web_link | Creating hyperlinks | **65.2** |
| | | web_form | Creating web forms | **52.2** |
| | | web_new | Using JavaScript "new" function | 23.3 |
| | | web_func | Creating JavaScript functions | 34.3 |
| | | web_behav | Creating web page behavior scripts | 42.9 |
| Databases | 79.3 | db_link | Referencing records by key | **74.0** |
| | | db_table | Creating tables | **71.7** |
| | | db_func | Creating stored procedures | 49.5 |
| | | db_view | Creating database views | **68.1** |
| Spreadsheets | 93.1 | ss_chart | Creating charts in spreadsheets | **80.1** |
| | | ss_func | Using functions ("sum") linking cells | **89.3** |
| | | ss_tmpl | Creating spreadsheet templates | 50.4 |
| | | ss_macro1 | Recording spreadsheet macros | 42.3 |
| | | ss_macro2 | Creating/editing spreadsheet macros | 38.6 |
| | | ss_link | Creating inter-spreadsheet references | **66.2** |
| Other Desktop Software | 96.1 | dsk_tmpl | Creating document templates | **73.3** |
| | | dsk_link | Making inter-document hyperlinks | **53.7** |
| | | dsk_macro1 | Recording desktop editor macros | 33.3 |
| | | dsk_macro2 | Creating/editing desktop editor macros | 30.8 |

Although the respondents were almost uniformly quite technically savvy, they did vary somewhat in their actual job function. Approximately 23% were IT or networking staff, 26% were IT or networking management, 17% were consultants or freelancers, and the remaining 35% were business staff or management. Moreover, respondents varied widely in their number of subordinates. Approximately 24% had no subordinates, another 29% had 1, 2, or 3 subordinates, another 28% had 4 to 10 subordinates (inclusive), and the remaining 19% had 11 or more subordinates.[4]

[4] Our survey pre-testers consistently interpreted the term "subordinate" to include the entire subtree of people beneath them, rather than just people who directly reported to them.

Just as our respondents varied widely in their job title and number of subordinates, they worked for firms that varied in total employee count and revenue. Specifically, approximately 44% of respondents worked for small firms with fewer than 100 workers, another 35% worked for medium firms numbering 100 to 4999 workers, and the remaining 21% worked for large firms numbering at least 5000 workers. Approximately 49% of firms had revenue under $50 million, another 40% had revenue of more than $50 million, and the remaining 11% chose not to report their company revenue.

Finally, although 10% of firms were IT vendors or related, the remainder were spread out over a wide variety of other industries. The most common was Consulting and Business Services at 17%, and the other top three after IT Vendors were Education (9%), Government (7%), and Non-Computer Industrial Manufacturing (7%). In short, the stereotypical respondent would probably be a member of the IT staff at a firm where IT does not comprise the firm's primary product or service.

### 3.4.2 Potential Non-Response Effects

*Information Week* had collected certain demographic variables in December 2004 to characterize the database from which they randomly selected the sample. In particular, we could compare our data set to the sample frame on firm industry, total employee count, and annual revenue. We found some differences between the two, particularly a relatively higher response rate by consultants and other non-managers, as well as people at small firms.

After survey collection had begun, we were disappointed to learn that the answer wording did not precisely match perfectly between the *Information Week* database demographic questions and the corresponding questions in our survey. Hence, most comparisons between the sample frame and the respondents must be done on a slightly qualitative basis.

Compared to members of the sample frame, our respondents were more likely to be in the Consulting and Business Services industry, as shown in Table 2. However, it appears that no single industry was very under-represented in our database compared to the sample frame. In particular, our sample contained about the same fraction of respondents in the IT industry as did the sample frame.

**Table 2: We found that Consultants and Business Services workers were slightly over-represented in our sample, compared to the sample frame. The most five common industries in our sample are shown below (N=816).**

| Industry | Our Sample | Sample Frame |
|---|---|---|
| Consulting and Business Services | 17% | 7% |
| IT Vendors | 10% | 10% |
| Education | 9% | 7% |
| Government | 7% | 9% |
| Non-Computer Manufacturing | 7% | 10% |

Compared to members of the sample frame, our respondents were somewhat more likely to work for small firms, as shown in Table 3. Fortunately, as noted in Section 5.2, firm size did not turn out to be a strong predictor of propensity to use features, so these differences in firm size caused by non-response are not likely to cause major problems in our analysis.

**Table 3: We found that small companies were slightly over-represented in our sample, compared to the sample frame. The relative proportions of small, medium, and large businesses are shown below (N=816).**

| Total Employee Count | Our Sample | Sample Frame |
|---|---|---|
| Fewer than 100 workers | 44% | 31% |
| 100 to 4999 workers | 35% | 33% |
| 5000 or more workers | 21% | 35% |

Compared to members of the sample frame, our respondents were more evenly divided between IT and non-IT jobs, and between management and non-management, as shown in Table 4. These differences might stem from wording differences between items in our survey and the *Information Week* database; spe-

cifically, whereas we refer to "IT or networking," the *Information Week* database refers to "Information Systems." However, it seems more likely that the managers who were contacted relatively less likely to have time or interest in answering our survey, resulting in a relatively lower response rate from management. Note also that *Information Week* intentionally cultivates their database in order to skew it toward managers and other "decision makers," so it is possible that this differential non-response fortuitously may have made our respondents *more* representative of a random American workplace, though it is difficult to know this for certain.

**Table 4: We found that our sample was more evenly divided between IT and non-IT in our sample, compared to the sample frame. The relative proportions of four occupational groups are shown below (N=816).**

| Total Employee Count | Our Sample | Sample Frame |
|---|---|---|
| IT or networking staff | 23% | 10% |
| IT or networking management | 26% | 54% |
| Consultant or freelancer | 17% | 7% |
| Business staff, management, and other | 35% | 28% |

# 4. Result 1: Identification of Three User Categories

In this section, we review the factor analysis statistical technique and then describe how we applied it using SAS 9.1 to our data. This analysis identified three factors underlying the covariance among users' propensity to use various programming-related features. These scales correspond to Macro, Linked-Structure, and Imperative programming.

As discussed below, the "best" set of factors is not uniquely defined—there are an infinite number of "rotations" of the factors in a multi-dimensional space that are equally valid ways of representing sources of covariance among data items. One way to verify that our factors are reasonable is to repeat the factor analysis with different settings and then to verify that the same qualitative factor structure results. We have performed this verification, as discussed in Section 4.4, and have found that our qualitative factor structure do not vary significantly as we vary our settings. Of course, these settings do produce slightly different *quantitative* factor loadings. It is the *qualitative* structure of the factors that proves robust under such a verification process; therefore, in Section 4, we largely focus on the *qualitative* structure (though for reference, we present the quantitative results of various factor analysis settings in Table 6 and Table 7).

Another way to check the quality of each factor generated by factor analysis is to use the factor's qualitative structure to guide the creation of a traditional scale whose Cronbach alpha can be calculated in the usual manner, as we discuss in Section 5.1.

## *4.1 Review of Factor Analysis*

In this section, we briefly review the goals and assumptions implicit in factor analysis. For further details, consult the introductory work [12] or the slightly more advanced [9] and [11].

### 4.1.1 Overview

A "factor" corresponds to an underlying variable that has not been measured but which is presumed to represent a construct responsible for causing covariant relationships among observed variables. That is, factor analysis aims to identify the reasons that observed variables "go together." Each factor loading is a vector each of whose cells reflects the correlation of that factor with one observed variable; thus, if a factor contains large values in two cells, then the corresponding two data variables each have a high correlation with the factor (and with each other).

Because factor analysis tends to highlight co-occurrence among data items, it is useful for guiding the creation of traditional scales. Creating a scale usually involves averaging together variables that measure a certain phenomenon or characteristic but do so with slightly varying emphases. Averaging the various variables helps to give a fuller picture of the underlying characteristic by "smoothing over" any bias in each particular variable.

For example, ethnography might record the amount of time that programmers spend in sixty different activities, ranging from typing code to talking on the phone. Factor analysis could potentially reveal

underlying factors that cause certain activities to co-occur. For instance, it might (hypothetically) reveal a "collaborative-ness" factor showing that people who often talk on the phone are also likely to perform other collaboration activities, such as writing emails and attending meetings. If researchers had previously theorized that such an underlying factor should exist, then the factor analysis lends credence to the theory, and the researchers would then be more justified in combining these items together into a scale representing the theorized "collaborative-ness" construct.

Note that detecting factors' underlying covariance is just the first step toward the ultimate goal of providing better tools. The next step is to understand *why* those factors exist; in the example above, for example, the collaborative-ness could correspond to an extroverted, communicative personality type or it could correspond to a business process requiring several collaborative activities. Our current survey and analysis aims at the first step, detecting factors that underlie co-occurrence, and in our future work, we will attempt to ferret out the reasons why these factors exist.

When performing factor analysis, it is important to avoid trying to extract more factors than the data will support. Each factor's loading vector is an eigenvector of the data's covariance matrix, and one commonly practiced threshold for accepting a factor is whether the corresponding eigenvalue exceeds 1.0. Moreover, there should usually be at least three observed variables for every factor to be extracted, so in the example of sixty activity measurements per respondent above, it would make no sense to try extracting more than twenty underlying factors.

In addition, exploratory factor analysis attempts to uncover factors explaining covariance among observed variables. If a variable lacks significant communality with the rest of the variables, then the outlier should not be included in the factor analysis, since doing so may lead to the appearance of spurious factor solutions. Moreover, writers emphasize the importance of having a theory or at least a conceptual framework, such as that described in Section 3, to guide the selection of variables to include in factor analysis.

### 4.1.2 Comparison to Principal Component Analysis

Factor analysis attempts to reveal a few underlying variables that explain aspects of many observed variables, and in this sense it resembles principal component analysis. However, these two techniques differ slightly in philosophy and assumptions. Principal component analysis attempts to represent a large number of observed variables as strict linear combinations of unmeasured underlying variables; it attempts to directly model observed variable values. In contrast, factor analysis does not attempt to predict variable *values*, but rather variable *covariance*. (In a sense, factor analysis is like a primary component analysis on the data's correlation matrix, rather than on the data itself.)

Consequently, principal component analysis assumes that observed variables can be completely modeled by these underlying variables, and if any variance remains unexplained, that this is simply because not all of the underlying variables have been identified as yet. In contrast, while factor analysis looks for "common" factors that explain covariance among multiple observed variables, it also recognizes that each observed variable may also have individual character, as well, and that each variable's variance is therefore the result of "unique" factors as well as "common" factors.

Because we seek to reveal co-occurrence patterns among feature usage propensities, we have selected factor analysis. As discussed in the following sections, this allows us to test our hypotheses about which features should "go together," and then to build scales representing those constructs.

## 4.2 Data Preparation & Adjusted Feature Propensity Items (AFPIs)

As described in Section 3.2, our survey asked about software usage by respondents and their subordinates. For each piece of software used, we asked about usage of features related to programming. We then performed the three steps described below to prepare the feature data for factor analysis.

First, we numerically coded each response as follows: "No" = 0, "Yes" = 1, "Don't Know" or skipped/missing = .U (which is a special symbol used by SAS to denote data that should not be included in analyses). Note that if the respondent did not report usage of the corresponding piece of software, then the feature variable was recorded as skipped/missing.

Second, we recognized that not every "Yes" is equally "resounding" or significant. For example, 66% of all respondents reported usage of web page editors and also creation of hyperlinks (item web_link),

whereas only 16% of all respondents reported usage of web server scripting and also coding of static server-side includes (item svr_include1). Consequently, we wanted to rescale answers so that a "Yes" for a more commonly used feature corresponded to less than 1.0, while a "Yes" for a less commonly used feature corresponded to more than 1.0. SAS includes the "proc standard" command precisely for this purpose, which standardized each variable to a mean of 0.0 and standard deviation of 1.0.

Finally, in preparation for revealing feature usage propensities, we needed to remove the effect that bundling has on feature usage. Obviously, feature usage tends to co-occur a great deal within the same tool: creation of hyperlinks, for example, is highly correlated with creation of web forms; such an effect has nothing to do with abstraction, but rather with the environment itself. To remove the effect of feature bundling within each tool, for each feature, we subtracted off the average feature usage by that respondent for that tool. For example, we adjusted each of the four database feature items for each respondent with equations like those shown in Table 5.

**Table 5: As exemplified by the database items below, we subtracted the average feature usage within the tool so that feature usage items better represent *relative* propensities.**

| Variable | Question Theme | Remove Tool Bundling Effect by Replacing the Variable's Value with the Value Shown Below |
|---|---|---|
| db_table | Creating tables | db_table – sum(db_table,db_view,db_func,db_link)/4 |
| db_view | Creating views | db_view – sum(db_table,db_view,db_func,db_link)/4 |
| db_func | Creating stored procedures | db_func – sum(db_table,db_view,db_func,db_link)/4 |
| db_link | Referencing records by key | db_link – sum(db_table,db_view,db_func,db_link)/4 |

By subtracting a baseline feature usage for each tool for each respondent, our adjustments reveal latent feature propensities that do not stem from the tool itself. The side-effect is that for each tool for each respondent, the adjusted feature propensity items must now total zero (e.g.: db_table + db_view + db_func + db_link now equals 0 in the example above). As a result, for each tool for each respondent, some of the adjusted feature propensity items will be positive, and some will be negative (or they will all be zero!), reflecting high or low feature usage propensities, respectively. We call the resulting 23 items "adjusted feature propensity items" (AFPIs) in the sections that follow. (See Table 6 for a list of the AFPIs.)

## 4.3 Characterization of User Categories

Given the adjusted feature propensity items (AFPIs) described in Section 4.2, we now turn to the question of which AFPIs are correlated with other AFPIs—which feature propensities co-occur? To answer this question, we applied factor analysis.

### 4.3.1 Factor analysis procedure

Unfortunately, applying factor analysis is not quite as straightforward as simply dumping the data into the SAS "proc factor" routine, which runs an exploratory factor analysis. As noted in Section 4.1, factor analysis relies on the assumption that a handful of common factors underlie the covariance among multiple observed variables. In conformance with this assumption, we had to ensure that we did not try to extract more factors than the data supported, and that we did not try to explain covariance between variables that did not share any covariance with the majority of the variables. Thus, the first few steps of factor analysis involve *identifying and removing* variables that do not share significant covariance with the remaining variables.

We started by running "proc factor" on all 23 AFPIs shown in Table 6 to see how many factors the data would support. Only three eigenvalues exceeded 1.0, so we reran the analysis with the restriction that SAS should attempt to model the covariance matrix using only three factors. (Retaining a fourth produced a relatively weakly loaded factor with little readily interpretable semantic content, which was not surprising since our first three factors together already explained 71% of the total variance.)

We then checked how much commonality the AFPIs shared with one another and found that 4 AFPIs had very low commonality with the other 19 AFPIs. Specifically, the sum of each AFPI's squared factor loadings was under 0.1, whereas the commonality for the other 19 AFPIs exceeded 0.1 and averaged over 0.4. We thus eliminated the following 4 AFPIs from further factor analysis:

- Creation of behaviors in web page editors (web_behav)
- Creation of functions/procedures in web server script (svr_func)
- Creation of data reference links between spreadsheets (ss_link)
- Creation of database views (db_view)

(Variables with low commonality such as 0.1 generally do not have large factor loadings, anyway, so retaining or omitting these variables does not change the results substantively.)

Once we removed variables that lacked significant covariance with the main body of variables, we ran our factor analysis with the remaining 19 AFPIs, yielding the three factor loadings shown in Table 6. These factors dealt with macros, linked structures, and imperative coding. The SAS command "proc factor method=p priors=max rotate=varimax", though running the analysis with other settings resulted in a similar qualitative pattern. Section 4.4 discusses the meaning of these parameters, and the effects of varying them.

**Table 6: Factor analysis of our data yielded the factor loadings shown below (N=168). Bolded cell values are somewhat large versus other items in that column; unbolded out numbers are relatively small. As discussed in the text, the four items at the bottom were omitted from our final factor loadings due to poor commonality.**

| AFPI Variable | Question Theme | Factor1 Macros | Factor2 LinkStruct | Factor3 Imperative |
|---|---|---|---|---|
| web_link | Creating hyperlinks | -0.05 | 0.06 | -0.62 |
| web_form | Creating web forms | 0.15 | **0.18** | -0.50 |
| web_new | Using JavaScript "new" function | -0.03 | -0.07 | **0.64** |
| web_func | Creating JavaScript functions | -0.04 | -0.09 | **0.50** |
| svr_include1 | Using Web server-side static includes | -0.09 | **0.25** | -0.26 |
| svr_include2 | Referencing PHP/Perl libraries | -0.06 | -0.41 | -0.06 |
| svr_new | Using PHP/Perl "new" function | 0.07 | 0.01 | **0.42** |
| ss_chart | Creating charts in spreadsheets | -0.40 | **0.27** | 0.14 |
| ss_func | Using functions ("sum") linking cells | -0.21 | **0.37** | -0.10 |
| ss_tmpl | Creating spreadsheet templates | -0.24 | -0.19 | 0.12 |
| ss_macro1 | Recording spreadsheet macros | **0.60** | -0.35 | -0.13 |
| ss_macro2 | Creating/editing spreadsheet macros | **0.68** | -0.18 | 0.02 |
| dsk_tmpl | Creating document templates | -0.59 | -0.02 | -0.17 |
| dsk_link | Making inter-document hyperlinks | -0.63 | -0.04 | 0.07 |
| dsk_macro1 | Recording desktop editor macros | **0.86** | 0.03 | 0.07 |
| dsk_macro2 | Creating/editing desktop editor macros | **0.87** | 0.07 | 0.09 |
| db_link | Referencing records by key | -0.02 | **0.68** | -0.06 |
| db_table | Creating tables | -0.01 | **0.54** | -0.21 |
| db_func | Creating stored procedures | 0.04 | -0.72 | **0.22** |
| web_behav | Creating web page behavior scripts | OMITTED | OMITTED | OMITTED |
| svr_func | Creating web server script functions | OMITTED | OMITTED | OMITTED |
| ss_link | Creating inter-spreadsheet references | OMITTED | OMITTED | OMITTED |
| db_view | Creating database views | OMITTED | OMITTED | OMITTED |

We used 168 respondent records in this factor analysis, rather than the full set of 816, due to limitations in the factor analysis technique, which cannot handle missing values. If respondents indicated that they and their respondents did not use a software tool, then our skip logic bypassed the questions corresponding to features within that tool. As a result, unless respondents indicated usage of all five software tools, we were missing answers for some or all feature questions, so we could not construct corresponding AFPIs. Thus, we only included 168 respondents in our initial factor analysis.

This is a serious limitation. It implies that the results of the factor analysis apply to only respondents who used every feature. To deal with this limitation, we will focus on factors' qualitative structure in Sections 4.3.2 and 4.4, then use that structure in Section 5 to guide the creation of traditional scales, which apply to *all* respondent records and checked using traditional measures such as Cronbach alpha.

### 4.3.2 Qualitative factor structure

By inspecting which AFPIs were most strongly correlated with each factor (the factor loadings in Table 6), we associated an apparent meaning with each factor.

The first factor corresponded to a cluster of features related to macros. Specifically, it correlated most positively with recording of spreadsheet macros (ss_macro1), textual editing of spreadsheet macros (ss_macro2), recording of macros in word processors and slide presentation editors (dsk_macro1), and textual editing of macros in word processors and slide presentation editors (dsk_macro2). In addition, the first factor demonstrated a negative correlation with the creation of inter-document hyperlinks (web_link), and also with document templates in word processing and slide presentation editors (dsk_tmpl). In other words, this "Macros" factor represents the fact that when people demonstrate a propensity toward one macro feature, they demonstrate strong positive propensity toward other macro features. (These users also demonstrated a somewhat weaker dis-propensity toward creating templates and hyperlinks, perhaps simply because of the normalization described in Section 4.2, or perhaps because users who know how to use macros have an inclination to use macros in situations where other users might use templates). For this reason, we have termed this factor the "Macro" factor.

The second factor corresponded to a cluster of features related to creating and linking data structures. Specifically, it correlated most positively with the AFPIs for creating and linking database tables via keys (db_table and db_link). It also loaded somewhat on the creation of other structured objects: web forms (web_form) and charts (ss_chart), which puzzled us until we realized that charts usually depict lists of X and Y values—namely, data structures called "tuples". The factor also loaded positively on linking web pages through "static includes" (svr_include1) and linking spreadsheet cells using simple functions like "sum" (ss_func). It negatively correlated with creation of database stored procedures (db_func), which was instead part of the third factor's cluster. Because the factor correlated so strongly with this cluster of features related to creating and linking structures, we have termed it the "LinkStruct" factor.

The third factor corresponded to a cluster of features related to imperative coding. Specifically, the third factor loaded most heavily on usage of the "new" command in Perl and PHP in web server scripting (svr_new), usage of the "new" command in JavaScript (web_new), the creation of functions in JavaScript (web_func), and creation of stored procedure functions in databases (db_func). It was negatively correlated with creation of hyperlinks (web_link) and web forms (web_form). Because the third factor correlated with this cluster of features related to imperative programming, we have termed it the "Imperative" factor.

To understand the significance of these results, let us focus for a moment on just one feature. For example, consider creation of database stored procedures (db_func). This feature was *not* part of the second factor's cluster of features, which included creation of database tables (db_table) and database keys (db_link). Instead, this second factor demonstrated a very strong negative correlation with creating database stored procedures. Thus, the adjustment procedure discussed in Section 4.2 did not tend to produce all zeroes for these three database AFPIs, but rather produced an "odd-man-out" effect. The fact that stored procedure creation was the "odd man out" suggests that the propensity to create stored procedures may come from a different underlying factor than does the propensity to create database tables and link them together.

Moreover, the people with a propensity to create stored procedures also had a tendency to create JavaScript functions, to use the JavaScript "new" command, and to use the Perl or PHP "new" command. In other words, stored procedure creation tended to co-occur with usage of other features in the "Imperative" cluster.

Thus, the analysis above uncovered an interesting result: People with a propensity to use imperative coding techniques in JavaScript, Perl, and PHP are more likely to use stored procedures in databases.

To summarize, feature propensities demonstrated a clear clustering effect. Section 4.4 reviews the robustness of our results and Section 5 describes differences among the "Macro," "LinkStruct," and "Imperative" users in terms of their programming practices.

## 4.4 *Robustness of Qualitative Factor Structure*

For any given data set, there exists an infinite number of alternate factor loadings which describe the correlation structure equally well. These represent linear combinations corresponding to rotations in a multi-dimensional space and signify equally valid ways of interpreting the data. The exact *quantitative* factor loadings will differ, but if running a factor analysis under a variety of reasonable settings continually produces the same *qualitative* structure, then this implies that the observed qualitative structure is a sensible way to represent the factors that underlie correlations within the data set.

To explore some alternate factor loadings, we took advantage of alternate algorithms supported by SAS for extracting factors. We found that these approaches generally generated "Macros," "LinkedStruct," and "Imperative" factors, though the exact loading coefficients varied somewhat.

### 4.4.1 Robustness under alternate extraction methods

Our initial analysis used the Principal Factor Analysis algorithm (specified with the METHOD=P setting). This algorithm resembles running a principal component analysis on the correlation matrix, except that each row's 1.0 entries on the correlation matrix diagonal are replaced with an estimate of that item's commonality. After principal component analysis runs, extracting a set of factors, SAS re-estimates these commonalities, re-populates the diagonal entries and re-runs the analysis. This repeats until commonalities cease to change much. Our initial estimate of the commonalities used the maximum value from each row to populate each diagonal entry of the correlation matrix (specified with the PRIORS=MAX setting).

We tried out two other factor analysis methods, ALPHA and ULS, revealing roughly the same qualitative structure as that described in Section 4.3. Whereas Principal Factor Analysis attempts to explain as much variance as possible within the correlation matrix, the ALPHA and ULS use different goals to guide their factor extraction. For example, the ALPHA method attempts to maximize a widely used measure of scale reliability, Cronbach alpha, for each of the generated factor-item loadings. Thus, it was encouraging to see these alternate algorithms generate the same qualitative structure as did our initial analysis.

### 4.4.2 Robustness under alternate rotation

After extracting the factors, they are linearly re-combined by SAS in a way that helps to make them more interpretable. One way to visualize this is as a rotation of the factors (as vectors) through a multi-dimensional space so that they are more parallel to item axes (while still remaining consistent with the data). The factors start out orthogonal to one another, meaning that they exhibit zero covariance. The VARIMAX rotation that we used in our initial analysis preserved this factor orthogonality.

An alternate approach, termed PROMAX, follows the VARIMAX rotation with a selective stretching of the vector space so that factor vectors lie even more parallel with item axes. This means that factors lose their orthogonality, so they become correlated with one another (and consequently, factor loadings no longer precisely equal correlations between factors and items). In exchange, the factors become more interpretable as a result. Applying PROMAX to our factors still generated the same qualitative structure, in terms of where the factor loads fell most heavily.

### 4.4.3 Robustness upon exclusion of managers

In feature questions, we asked whether the respondent or any subordinates had used the features in the past three months. We wondered whether the same factors would appear if we repeated our initial analysis using only the 43 respondents who had no subordinates and who had no missing values in relevant questions.

As shown in Table 7, the overall qualitative structure of the factors remained the same as in our initial analysis: the first factor loaded on macro AFPIs, the second loaded on structure and linkage, and the third loaded on imperative programming. While the *qualitative* structure remained the same, some loadings strengthened slightly while others slightly weakened. Specifically, relative to our initial analysis, the most significant changes were increased loading by the third factor on referencing of PHP/Perl libraries (svr_include2) and using the PHP/Perl "new" function (svr_new), both of which are related to imperative web server scripting and are therefore consistent with the "Imperative" label we associated with the third factor in our initial analysis.

**Table 7: Factor analysis of data from respondents with no subordinates yielded the factor loadings shown below (N=43). Bolded cell values are somewhat large compared to other items in that column; unbolded numbers are relatively small.**

| AFPI Variable | Question Theme | Factor1 Macros | Factor2 LinkStruct | Factor3 Imperative |
|---|---|---|---|---|
| web_link | Creating hyperlinks | 0.03 | 0.03 | -0.48 |
| web_form | Creating web forms | 0.37 | **0.22** | -0.23 |
| web_new | Using JavaScript "new" function | -0.19 | -0.05 | **0.45** |
| web_func | Creating JavaScript functions | 0.05 | -0.22 | **0.17** |
| svr_include1 | Using Web server-side static includes | -0.07 | **0.12** | -0.59 |
| svr_include2 | Referencing PHP/Perl libraries | -0.16 | -0.10 | **0.53** |
| svr_new | Using PHP/Perl "new" function | 0.14 | -0.12 | **0.36** |
| ss_chart | Creating charts in spreadsheets | -0.47 | 0.08 | 0.06 |
| ss_func | Using functions ("sum") linking cells | 0.10 | **0.36** | -0.12 |
| ss_tmpl | Creating spreadsheet templates | -0.14 | -0.18 | **0.43** |
| ss_macro1 | Recording spreadsheet macros | **0.56** | -0.16 | -0.07 |
| ss_macro2 | Creating/editing spreadsheet macros | **0.61** | -0.03 | -0.07 |
| dsk_tmpl | Creating document templates | -0.66 | -0.15 | -0.38 |
| dsk_link | Making inter-document hyperlinks | -0.53 | -0.06 | **0.48** |
| dsk_macro1 | Recording desktop editor macros | **0.80** | **0.18** | -0.02 |
| dsk_macro2 | Creating/editing desktop editor macros | **0.86** | 0.10 | -0.08 |
| db_link | Referencing records by key | -0.07 | **0.97** | -0.08 |
| db_table | Creating tables | -0.07 | **0.97** | -0.08 |
| db_func | Creating stored procedures | -0.12 | -0.79 | **0.16** |
| web_behav | Creating web page behavior scripts | OMITTED | OMITTED | OMITTED |
| svr_func | Creating web server script functions | OMITTED | OMITTED | OMITTED |
| ss_link | Creating inter-spreadsheet references | OMITTED | OMITTED | OMITTED |
| db_view | Creating database views | OMITTED | OMITTED | OMITTED |

## 5. Result 2: Links between Abstraction, Testing, and Documentation

Guided by the qualitative structure discovered through our factor analysis, we proceeded to construct scales representing the "Macro," "LinkedStruct," and "Imperative" propensities of respondents. ANOVA tests revealed that high propensity to use Imperative features was associated with a tendency to test and document, while high propensity to use LinkStruct features was only weakly associated with these practices.

### *5.1 Definition of Factor-like Scales*

As noted in Section 4.3.1, SAS omits records with missing values in factor analysis. Consequently, guided by the factor loadings shown in Table 6, we wish to create traditional scales that can be applied to all respondents.

We were guided for two reasons by the *qualitative* factor structure shown in Table 6, rather than the particular correlation coefficients listed. First, the exact numbers varied somewhat depending on which factor analysis algorithm we ran, as discussed in Section 4.4. Second, we wanted to include only the most significant items in each factor, rather than "throwing in the kitchen sink," in order to keep the semantics of each factor as interpretable as possible. Thus, for each of the three factors, we chose to sort the items according to absolute value of correlation coefficient, and then include in each scale only the five or six items with the largest absolute value of correlation coefficient.

For example, as shown in Table 6, the "Macros" factor had a large positive correlation with four items and a large negative correlation with two items. Specifically, it correlated positively with the recording of spreadsheet macros (ss_macro1), the textual editing of spreadsheet macros (ss_macro2), the recording of macros in word processors and slide presentation editors (dsk_macro1), and the textual editing of macros in word processors and slide presentation editors (dsk_macro2). The factor also had a strong correlation, but with a *negative* coefficient, with the creation of templates in word processors and slide presentation editors (dsk_tmpl) and the creation of links among documents created in word processors and slide presentation

editors (dsk_link). (Negative values are a side-effect of the AFPI adjustment procedure outlined in Section 4.2, and negative coefficients simply result from negative correlations.) Hence, we include these six variables (ss_macro1, ss_macro2, dsk_macro1, dsk_macro2, dsk_tmpl, and dsk_link) in our "Macros" scale. Note that the last two of these will be multiplied by a coefficient of -1 because they were *negatively* correlated with the "Macros" factor.

Based on the factor loadings' qualitative structure, we defined the following scales, which had standardized Cronbach alpha values of 0.82, 0.62, and 0.64, respectively:

- PX_MACROS = (ss_macro1 + ss_macro2 + dsk_macro1 + dsk_macro2 – dsk_tmpl – dsk_link)/6.0;
- PX_LINKSTRUCT = (web_form + svr_include1 + db_link + db_table – db_func)/5.0;
- PX_IMPERATIVE = (web_new + web_func + svr_new – web_link – web_form)/5.0;

In the case of missing values, we omitted the corresponding adjusted feature propensity item (AFPI) from the formulas above, and adjusted the divisor accordingly. After applying these scales to each respondent, we standardized each scale using "proc standard" to a mean of 0 and standard deviation of 1. The resulting scales were not normally distributed (failing a Kolmogorov-Smirnov test at P=0.01, N=816), but they were essentially skewed unimodal with no floor or ceiling effects.

## 5.2 Comparisons among User Categories

In order to assess differences among categories of users, we created several scales (shown in Table 8) representing concepts related to programming and general computing skills. Missing variables in scales were replaced with 0 ("No" or "Never Useful"), then variables were standardized and averaged to generate the scale. (There was no effect on the ANOVA test results discussed below if we omitted records with missing values, rather than replacing with zero, or if we omitted respondents with subordinates.)

After we had defined the three factor-like scales discussed in Section 5.1, as well as the scales shown in Table 8, we could determine whether our factor-like scales gave any "predictive power" with respect to the other scales. To make this assessment, we divided the respondents into two categories—those who scored above average on the Macro scale, and those that scored below average on the Macro scale—and then used a standard ANOVA test to see whether respondents in each category differed on a scale of interest, such as knowledge of programming terms. We repeated this analysis likewise for the LinkStruct and Imperative scales.

As shown in Table 9, our factor-like scales did yield significant predictive power for many scales of interest. For example, respondents with greater-than-average propensity to use Imperative features had a higher knowledge of programming terms than did respondents with lower-than-average propensity to use Imperative features. In fact, people with higher propensity to use Imperative features showed higher sophistication on every "good programming practices" scale, including testing and documenting. Compared to people with a low propensity to use Imperative features, people with high propensity to use Imperative features reported more knowledge of programming terms, more usage of programming constructs, more web page testing, more spreadsheet testing, more propensity to create documentation, and higher perceived usefulness of the web. ($P \leq 0.02$ for these results. Non-empty cells in Table 9 show statistically significant Type I ANOVA tests of the row's specified scale based on the column's specified categorization; all indicated cells were also significant under Tukey's test. The three categorizations were tested independently with df=1, N=816. No other cells were significant at the P=0.05 level. The shown P values are conservative, in that they incorporate Bonferroni's correction to help control Type I error.)

Likewise, people with higher-than-average propensity to use LinkStruct features were somewhat more sophisticated than people with lower-than-average propensity to use LinkStruct features. Specifically, they reported more knowledge of programming terms, more usage of programming constructs, and more web page testing. ($P \leq 0.03$ for these results.)

Finally, people with higher-than-average propensity to use Macro features were more likely to report usage of programming constructs than were people with lower-than-average propensity to use Macro features. ($P \leq 0.02$ for these results.)

**Table 8: We defined the scales shown below to assess differences among categories of users (N=816).**

| Scale | Variable | Question Theme |
|---|---|---|
| Knowledge of programming terms Cronbach alpha = 0.84 | know_variable | Personal familiarity with variables |
| | know_subroutine | Personal familiarity with subroutines |
| | know_conditional | Personal familiarity with conditionals |
| | know_loop | Personal familiarity with loops |
| Usage of programming constructs Cronbach alpha = 0.89 | know_variable_more | Personal usage of variables |
| | know_subroutine_more1 | Personal usage of others' subroutines |
| | know_subroutine_more2 | Personal creation of subroutines |
| | know_conditional_more | Personal usage of conditionals |
| | know_loop_more | Personal usage of loops |
| Web page testing (single item) | web_test | Testing web pages in many browsers |
| Spreadsheet testing (single item) | ss_test | Testing spreadsheets for accuracy |
| Propensity to create documentation Cronbach alpha = 0.71 | ss_doc | Creates documentation accompanying spreadsheets |
| | db_doc | Creating documentation accompanying databases |
| | www_dest_ex | Perceived usefulness of web as a place to publish examples |
| | www_dest_doc | Perceived usefulness of web as a place to publish documentation |
| | www_dest_data | Perceived usefulness of web as a place to publish numerical data |
| Perceived usefulness of the web Cronbach alpha = 0.82 | www_src_ex | Perceived usefulness of web as a source of examples |
| | www_src_doc | Perceived usefulness of web as a source of documentation |
| | www_src_data | Perceived usefulness of web as a source of numerical data |
| | www_dest_ex | Perceived usefulness of web as a place to publish examples |
| | www_dest_doc | Perceived usefulness of web as a place to publish documentation |
| | www_dest_data | Perceived usefulness of web as a place to publish numerical data |
| On-the-job training Mean of non-missing variables Cronbach alpha = 0.76 | demo_train_job_prog | Training in programming |
| | demo_train_job_web | Training in creating web pages |
| | demo_train_job_ss | Training in creating spreadsheets |
| | demo_train_job_db | Training in creating databases |
| On-the-job training in programming (single item) | demo_train_job_prog | Training on-the-job in programming |
| High school training (single item) | demo_train_hs | High school training in programming |
| Education (single item) | demo_education | Highest level of education |
| Job function (single item) | demo_job | Job function: IT Staff / IT Mgmt / Consultant / Other |
| College major (single item) | demo_major1 | Major: Computer-centric / Math-related (e.g.: sciences) / Other |
| Firm Industry (single item) | demo_industry | Industry: IT Vendor / Other |
| Firm Revenue (single item) | demo_revenue | Firm yearly revenue |
| Firm Size (single item) | demo_empcount | Firm total employee count |

**Table 9: As shown below, we found a number of statistically significant differences among categories of users. (df=1, N=816.)**

| Scale | Categorize based On Macros | Categorize based On LinkStruct | Categorize based On Imperative |
|---|---|---|---|
| Knowledge of programming terms | | P = 0.03; F = 4.82 | P < 0.01; F = 11.52 |
| Usage of programming constructs | P < 0.01; F =7.25 | P < 0.01; F = 8.27 | P < 0.01; F = 16.17 |
| Web page testing | | P < 0.01; F = 16.34 | P < 0.01; F = 87.76 |
| Spreadsheet testing | | | P = 0.02; F = 5.14 |
| Propensity to create documentation | | | P < 0.01; F = 81.49 |
| Perceived usefulness of the web | | | P < 0.01; F = 56.47 |
| On-the-job training | | | |
| On-the-job training in programming | | | |
| High school training programming | | | |
| Education | | | |
| Job function | P = 0.02; F =5.27 | | P < 0.01; F = 7.44 |
| College major | | | |
| Firm Industry | | | P < 0.01; F = 7.79 |
| Firm Revenue | | | |
| Firm Size | | | |

Looking at the results in Table 9 from a qualitative standpoint, it is clear that a person's propensity to use Imperative features is strongly related to what we might expect from a "good" programmer—knowledge of terminology, testing, and documentation. This statement is true to a lesser extent about propensity to use LinkStruct features, which was associated with knowledge of terminology and some testing practice, but not associated with documentation. Finally, of the three factor-like scales, propensity to use Macros seems to be least strongly associated with "good" programming.

As shown in Table 9, the respondents in each category generally did not differ significantly with respect to demographics. Regardless of which factor-like scale is used to categorize users, categories did not differ statistically on most demographic items. In particular, training, education, and firm size did not differ markedly from one category of user to the other.

The main exception is that people with a strong propensity to use Imperative features were more likely to have IT jobs and to work in the IT industry. This is not surprising, considering the heavy use of imperative languages by professional programmers in the IT industry. The data also weakly suggest that people with a stronger propensity to use Macros were more likely to have IT jobs.

An analysis like this yields correlative results but not causative conclusions: several hypotheses might explain these patterns. For example, Imperative features might be so difficult to use that learning terminology, testing, and documentation is a necessary precondition; LinkStruct and Macros features simply might be easier to use. Alternatively, perhaps people who use Imperative features work in environments where testing and documentation are demanded by management or the corporate culture. Perhaps the true cause is a mix of these, or a third cause altogether. Further study will be required to ferret out the motivations, background, and skills of people who use each class of features.

In short, although our factor-like scales are strongly related to the "good programming" scales discussed earlier, our factor-like scales are generally not as strongly related to demographic variables.

## 6. Conclusion and Future Work

Our survey of *Information Week* readers has revealed significant clustering of propensities to use features related to the creation of abstractions. We have developed three scales representing these clusters, corresponding to the use of Macro features, Linked Structure features, and Imperative features. Moreover, we have found that respondents higher on these scales tend to be more likely to report testing and documenting, and this effect is most pronounced along the Imperative scale.

These three scales essentially represent dimensions spanning the space of end user programming in tech-savvy workplaces, and surveys such as ours reveal patterns within this space. However, there are limitations to what can be accomplished with a survey.

For example, this survey is limited in the generalizability of its results. It is not clear that end users in home or school contexts would demonstrate the same patterns of feature use. Indeed, our earlier secondary analysis of government data suggests that end users in these other contexts have significantly different application usage than users in the workplace context [25], and previous ethnographies suggest that the menu of abstractions available to end users in the home context also differ from that of the workplace context [20]. Thus, mapping out the space end user programming outside the workplace would require additional surveys.

In addition, although surveys can reveal patterns and trends within a population, they often offer limited insight into why those patterns and trends exist. In our case, it is not yet clear why feature usage propensities demonstrate the clustering that we observed, and possible hypotheses abound. Addressing this issue further would require a more interactive study that spans multiple types of user and multiple application environments. For example, we might wish to perform a contextual inquiry of users from several occupations and observe how they go about using features in different applications to create abstractions.

We are particularly interested in understanding use of features related to Linked Structure abstractions. It is clear that Imperative programming strongly resembles the work of a professional programmer, so it is not too surprising that respondents with a propensity for using Imperative features would demonstrate various good programming practices such as testing and documenting (particularly since our sample included a significant number of users in IT-related jobs or industries).

But in this modern world of relational databases and object-oriented programming, we might also suppose that Linked Structure programming *also* resembles the work of a professional programmer. Why, then, was the propensity to create Linked Structures only *weakly* related to testing and documenting (compared to propensity to use Imperative features)? Does the difference a property of the users themselves? Does the difference stem from the business processes surrounding the creation of those abstractions? Or is it just easier to perform testing and documenting of Imperative programs?

We intend to focus on these questions and others in our future work as we continue to identify and study different categories of end users, guided by the hints from this survey.

## 7. Acknowledgements

We gratefully acknowledge Rusty Weston and Lisa Smith at *Information Week* for their hard work suggesting refinements for the survey, publishing it on the web, advertising it to their subscribers, and sharing the data thus collected. We thank the subscribers of *Information Week* magazine for participating in this study. Finally, we extend our gratitude to James Herbsleb, Irina Shklovski, and Sara Kiesler for their input on various aspects of this study: how to develop a model, how to clean and analyze data, and how to design a survey, respectively.

This work has been funded in part by the EUSES Consortium via the National Science Foundation (ITR-0325273), by the National Science Foundation under Grant CCF-0438929, by the Sloan Software Industry Center at Carnegie Mellon, and by the High Dependability Computing Program from NASA Ames cooperative agreement NCC-2-1298. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## 8. References


[1] L. Beckwith, M. Burnett, S. Wiedenbeck, and C. Cook. Effectiveness of End-User Debugging Software Features: Are There Gender Issues?. *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, 2005, pp. 869-878.

[2] A. Blackwell. First Steps in Programming: A Rationale for Attention Investment Models. *HCC '02: Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, IEEE Computer Society, 2002, pp. 2-10.

[3] N. Edwards. The Effect of Certain Modular Design Principles on Testability. *Proceedings of the International Conference on Reliable Software*, 1975, pp. 401-410.

[4] M. Fisher II and G. Rothermel. The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. Technical Report 04-12-03, University of Nebraska-Lincoln, Lincoln, NE, December 2004.

[5] T. Green and A. Blackwell. Ironies of Abstraction. *Proceedings of the 3rd International Conference on Thinking, British Psychological Society*, 1996.

[6] T. Green and M. Petre. Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework. *Journal of Visual Languages and Computing*, Vol. 7, No. 2, June 1996, pp. 131-174.

[7] J. Hintze and M. Masuch. Designing a 3D Authoring Tool for Children. *C5 '04: Proceedings of the Second International Conference on Creating, Connecting and Collaborating Through Computing*, IEEE Computer Society, 2004, pp. 78-85.

[8] Inquisite web survey system. http://www.inquisite.com

[9] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[10] K. Kahn. Drawings on Napkins, Video-Game Animation, and Other Ways To Program Computers. *Communications of ACM*, Vol. 39, No. 8, 1996, pp. 49-59.

[11] J. Kim and C. Mueller. *Factor Analysis: Statistical Methods and Practical Issues*, Sage Publications, Beverly Hills, CA, 1978.

[12] J. Kim and C. Mueller. *Introduction to Factor Analysis: What It Is and How to Do It*, Sage Publications, Beverly Hills, CA, 1978.

[13] R. Miller and B. Myers. Multiple Selections in Smart Text Editing. *IUI '02: Proceedings of the 7th International Conference on Intelligent User Interfaces*, ACM Press, 2002, pp. 103-110.

[14] F. Modugno, A. Corbett, and B. Myers. Evaluating Program Representation in a Demonstrational Visual Shell. *CHI '95: Conference Companion on Human Factors in Computing Systems*, ACM Press, 1995, pp. 234-235.

[15] B. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*, The MIT Press, Cambridge, MA, 1993.

[16] J. Pane, B. Myers, and L. Miller. Using HCI Techniques to Design a More Usable Programming System. *HCC '02: Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, IEEE Computer Society, 2002, pp. 198-206.

[17] J. Pane and B. Myers. Usability issues in the design of novice programming systems. Technical Report CMU-CS-96-132, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1996.

[18] D. Parnas. The Influence of Software Structure on Reliability. In *Proceedings of the International Conference on Reliable Software*, 1975, pp. 358-362.

[19] D. Parnas. On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15, 12 (Dec. 1972), 1053-1058.

[20] J. Rode, E. Toye, and A. Blackwell. The Fuzzy Felt Ethnography—Understanding the Programming Patterns of Domestic Appliances. In *Proceedings of the 2nd International Conference on Appliance Design.*, Vol. 8, No. 3-4, 2004, pp. 161-176.

[21] M. Rosson, J. Ballin, and J. Rode. Who, What, and How: A Survey of Informal and Professional Web Developers, *VL/HCC'05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2005, pp. 199-206.

[22] J. Ruthruff, E. Creswick, M. Burnett, and C. Cook. End-User Software Visualizations for Fault Localization. *SoftVis '03: Proceedings of the 2003 ACM Symposium on Software Visualization*, ACM Press, 2003, pp. 123-132.

[23] C. Scaffidi, M. Shaw, and B. Myers. The "55M End-User Programmers" Estimate Revisited. Technical Report CMU-ISRI-05-100/CMU-HCII-05-100, Institute for Software Research International, Carnegie Mellon University, Pittsburgh, PA, 2005.

[24] C. Scaffidi, M. Shaw, and B. Myers. An Approach for Categorizing End User Programmers to Guide Software Engineering Research. *WEUSE I: Proceedings of the First Workshop on End-User Software Engineering*, ACM Press, 2005, pp. 1-5.

[25] C. Scaffidi, M. Shaw, and B. Myers. Estimating the Numbers of End Users and End User Programmers. *VL/HCC'05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 207-214, 2005.

[26] C. Wallace, C. Cook, J. Summet, and M. Burnett. Assertions in End-User Software Engineering: A Think-Aloud Study. HCC '02: Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments, IEEE Computer Society, 2002, pp. 63.

[27] A. Wilson, et. al. Harnessing Curiosity to Increase Correctness in End-User Programming, *Proceedings of CHI'2003*, pp. 305-312.

## Appendix A

This Appendix shows the wording of the questions in the survey that we deployed with *Information Week*. To facilitate cross-referencing between these questions and our discussion in Sections 1 through 6, above, we have enclosed variable names in double angle brackets, below, for example: <<demo_empcount>>

Note that some questions were skipped. The skip logic is shown in italics and square brackets below, for example: *[Only ask this question if <<web>> = Yes]*

Finally, some questions were asked together on the same screen. For example, after asking about software usage, if the respondent reported usage of that piece of software, then the next screen listed a series of features and asked the respondent about usage of those features. To highlight the fact that these questions were asked all on one screen, we have enclosed them in boxes, below.

This is a study by Carnegie Mellon University that looks at the software that you use, and considers how software features help you do your job. Your answers will help guide research aimed at improving software flexibility. Your completed survey makes you eligible for a chance to win a grand prize check for $500 and one of five $100 checks from InformationWeek, part of CMP Media. Completing this survey also entitles you to a *free* report detailing the study results.

Please be assured that your responses will remain confidential. Answers will only be reported in aggregate and nothing will be attributed to you or your company. Study results will appear in an upcoming story in print and on www.informationweek.com in May.

Thank you again for participating in this voluntary survey.

**Section 1 of 4**
**This first section asks about the software that you and your subordinates use at work.**

<<web>> In the past 3 months, have you or your subordinates created or edited web pages (for example, using FrontPage, QuarkXPress, Photoshop, or DreamWeaver)?
o Yes
o No

***[Only ask this question if <<web>> = Yes]***
In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<web_link>>Created hyperlinks from one web page to another | Yes | No | Don't know |
| <<web_form>>Created web forms to collect information | Yes | No | Don't know |
| <<web_new>>Used the "new" command to create JavaScript or ActionScript objects | Yes | No | Don't know |
| <<web_behavior>>Used "behaviors" (such as mouse rollovers) to make web pages interactive | Yes | No | Don't know |
| <<web_func>>Created JavaScript or ActionScript functions that accept parameters | Yes | No | Don't know |
| <<web_test>>Created web pages and tested them in multiple different browsers | Yes | No | Don't know |

<<svr>> In the past 3 months, have you or your subordinates created or edited web server scripts (such as PHP, VBScript/ASP, or Perl)?

o Yes
o No

***[Only ask this question if <<svr>> = Yes]***

In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<svr_include1>>Used the "require" command in PHP or Perl pages to include other pages | Yes | No | Don't know |
| <<svr_include2>>Used the "include" command in VBScript or shtml pages to include other pages | Yes | No | Don't know |
| <<svr_form>>Created web forms to collect information | Yes | No | Don't know |
| <<svr_new>>Used the "new" command to create PHP or Perl objects, or the "CreateObject" command to create VBScript objects | Yes | No | Don't know |
| <<svr_func>>Created web server script functions or subroutines that accept parameters | Yes | No | Don't know |
| <<svr_test>>Created web pages and tested them in multiple different browsers | Yes | No | Don't know |
| <<svr_doc>>Inserted comments into web server scripts, such as PHP, Perl, or VBScript | Yes | No | Don't know |

<<ss>> In the past 3 months, have you or your subordinates created or edited spreadsheets (for example, using Excel or Lotus 1-2-3)?

o Yes
o No

***[Only ask this question if <<ss>> = Yes]***

In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<ss_doc>>Created documents to accompany a spreadsheet, explaining how to use the spreadsheet file | Yes | No | Don't know |
| <<ss_chart>>Used data in spreadsheet cells to create charts or graphs | Yes | No | Don't know |
| <<ss_func>>Used spreadsheet functions (such as "sum") when typing into cells | Yes | No | Don't know |
| <<ss_link>>Created data references to automatically incorporate data from one spreadsheet or workbook into another | Yes | No | Don't know |
| <<ss_test>>Tested spreadsheets for accuracy | Yes | No | Don't know |
| <<ss_tmpl>>Created spreadsheet templates | Yes | No | Don't know |
| <<ss_macro1>>Recorded macros | Yes | No | Don't know |
| <<ss_macro2>>Created or edited macros using the macro editor | Yes | No | Don't know |

<<dsk>> In the past 3 months, have you or your subordinates created or edited word processing or slide presentation documents (for example, using WordPerfect, PowerPoint, or Microsoft Word)?
o Yes
o No

***[Only ask this question if <<dsk>> = Yes]***
In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<dsk_tmpl>>Created word processing or slide presentation document templates | Yes | No | Don't know |
| <<dsk_link>>Created hyperlinks to link one word processing or slide presentation document to another | Yes | No | Don't know |
| <<dsk_macro1>>Recorded macros | Yes | No | Don't know |
| <<dsk_macro2>>Created or edited macros using the macro editor | Yes | No | Don't know |

<<db>> In the past 3 months, have you or your subordinates created or edited data in a database (for example, in FileMaker, Oracle, Sybase, SQL Server, or Access)?
o Yes
o No

***[Only ask this question if <<db>> = Yes]***
In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<db_link>>Referenced database records using a key field (such as social security number or other identifier) | Yes | No | Don't know |
| <<db_table>>Created a new database table, or added columns to an existing database table | Yes | No | Don't know |
| <<db_view>>Created a new database view, or added columns to an existing database view | Yes | No | Don't know |
| <<db_func>>Created database stored procedures | Yes | No | Don't know |
| <<db_doc>>Created documents to accompany a database, explaining how to use the database | Yes | No | Don't know |

<<bi>> In the past 3 months, have you or your subordinates created or edited custom business intelligence reports (for example, using Microsoft Analysis Services, Hyperion/Brio, Cognos/Adaytum, or Business Objects/Crystal)?
o Yes
o No

*[Only ask this question if <<bi>> = Yes]*
In the past 3 months, have you or your subordinates done any of the following?

| | | | |
|---|---|---|---|
| <<bi_subrep>>Incorporated subreports into a larger business intelligence report | Yes | No | Don't know |
| <<bi_link>>Joined database tables or views via key fields (such as social security number or other identifier) while constructing a business intelligence report | Yes | No | Don't know |
| <<bi_chart>>Used data in databases to create charts or graphs in a business intelligence report | Yes | No | Don't know |
| <<bi_olap>>Helped design the dimensions of an OLAP repository | Yes | No | Don't know |
| <<bi_sql>>Specified an SQL query to retrieve data from a database for use in a business intelligence report | Yes | No | Don't know |
| <<bi_filter>>Created a business intelligence report which allowed the person viewing the report to filter (hide / show) portions of the report | Yes | No | Don't know |
| <<bi_sort>>Created business intelligence reports which allowed the person viewing the report to change the sort order of information in the report | Yes | No | Don't know |
| <<bi_drill>>Created business intelligence reports which allowed the person viewing the report to "drill down" into more detailed report data | Yes | No | Don't know |
| <<bi_doc>>Created documents to accompany a business intelligence report, explaining how to interpret the report | Yes | No | Don't know |

<<backups>>In the past 3 months, have you or your subordinates backed up files by making extra copies, or having IS/IT staff do so?
o Yes
o No
o I don't know

**Section 2 of 4**
**The next few questions ask about the relationship between your work and the software you use. Note that these questions ask about the past 1 year.**

<<sw1>> <<sw2>> <<sw3>>List the three software applications that you or your subordinates have used the most in the past year.

<<sw_help>>How has this software helped get work done in the past year?

<<sw_hinder>>In what ways has this software "gotten in the way" of doing work in the past year?

<<sw_custom>> In the past year, have you or your subordinates customized this software? If so, how?

<<sw_useless>>In the past year, have you or your subordinates purchased software for work but then used it less than anticipated?
o Yes
o No

***[Only ask this question if <<sw_useless>> = Yes]***
<<sw_useless_more>>Why did you use this software less than anticipated?

***[Only ask this question if <<web>> = Yes OR <<svr>> = Yes]***

<<copy_web>>In the past year, when you or your subordinates created a new web page, did you get started by copying an existing web page?
o Usually
o Sometimes
o Rarely
o Never

***[Only ask this question if <<ss>> = Yes]***
<<copy_ss>>In the past year, when you or your subordinates created a new spreadsheet, did you get started by copying an existing spreadsheet?
o Usually
o Sometimes
o Rarely
o Never

***[Only ask this question if <<bi>> = Yes]***
<<copy_bi>>In the past year, when you or your subordinates created a new business intelligence report, did you get started by copying an existing report?
o Usually
o Sometimes
o Rarely
o Never

<<www_src_ex>>Consider how you or your subordinates used the web when you created or edited business intelligence reports, spreadsheets, web pages, and other documents.

In the past year, how useful has the web been for your work as a source of examples of reports, spreadsheets, web pages, or other documents?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<www_src_doc>> In the past year, how useful has the web been for your work as a source of documentation?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<www_src_data>> In the past year, how useful has the web been for your work as a source of numerical data?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<www_dest_ex>> In the past year, how useful has the web been for your work as a place to publish documents (reports, spreadsheets, web pages, or documents)?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<www_dest_doc>> In the past year, how useful has the web been for your work as a place to publish documentation (telling how to use or interpret your reports, spreadsheets, web pages, or documents)?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<www_dest_data>> In the past year, how useful has the web been for your work as a place to publish numerical data?
o Usually useful
o Sometimes useful
o Rarely useful
o Never useful

<<help>>When you or your subordinates have trouble creating web pages, spreadsheets, reports, or other documents, what is your main source of help?

<<help_more>>Why is this help preferable to other sources of help?

**Section 3 of 4**
**The next few questions ask about your programming knowledge and practice. Note that these questions ask about you personally, rather than about your subordinates.**

<<know_variable>>Are you personally familiar with the concept of a "variable"--a named memory location where programs can store intermediate results and/or read them back?
o Yes
o No

***[Only ask this question if <<know_variable>> = Yes]***
<<know_variable_more>>When creating programs in the past year, have you personally defined and referenced variables?
o Yes
o No

<<know_subroutine>>Are you personally familiar with the concept of a "subroutine"--a piece of code that can be called from multiple places in a program (also known as a "procedure", "function", or "method")?
o Yes
o No

***[Only ask this question if <<know_subroutine>> = Yes]***
<<know_subroutine_more1>> When creating programs in the past year, have you personally called a subroutine that another person in your organization already created?
o Yes
o No

***[Only ask this question if <<know_subroutine>> = Yes]***
<<know_subroutine_more2>> When creating programs in the past year, have you personally created a new subroutine?
o Yes
o No

<<know_conditional>>Are you personally familiar with the concept of a "conditional"--a command that prevents code from running unless a certain condition is met (also known as "if/then", "if/else", or "unless" commands)?
o Yes
o No

***[Only ask this question if <<know_conditional>> = Yes]***
<<know_conditional_more>> When creating programs in the past year, have you personally coded conditional statements?
o Yes
o No

<<know_loop>>Are you personally familiar with the concept of a "loop"--a sequence of instructions that the computer repeats, either until some condition is met, or indefinitely (also known as a "for/next", "while/do", or "repeat/until" command sequence)?
o Yes
o No

***[Only ask this question if <<know_loop>> = Yes]***
<<know_loop_more>> When creating programs in the past year, have you personally coded loops?
o Yes
o No

***[Only ask this question if <<know_variable>> = Yes OR <<know_subroutine>> = Yes OR <<know_conditional>> = Yes OR <<know_loop>> = Yes]***
<<comments>> <<comments_more>>Do you insert comments into your code? If so, what information do you write in your comments?

***[Only ask this question if <<know_variable>> = Yes OR <<know_subroutine>> = Yes OR <<know_conditional>> = Yes OR <<know_loop>> = Yes]***
<<documents>> <<documents_more>>Do you create separate documentation describing your code? If so, what information do you write in your documentation?

**Section 4 of 4**
**The last few questions ask about your background.**

<<demo_job>>Which of the following best describes your current job function?
o IT or networking staff
o IT or networking management
o Business unit staff
o Business management
o Student
o Consultant
o Retired
o Other <<demo_job_more>>

<<demo_revenue>>For research classification purposes only, what's the annual revenue (or operating budget) of your entire organization?
o Less than $6 million
o $6 million to $49.9 million
o $50 million to $99.9 million
o $100 million to $499.9 million
o $500 million to $999.9 million
o $1 billion to $4.9 billion
o $5 billion or more

<<demo_empcount>> How many employees are in your company?
o Less than 50
o 50-99
o 100-499
o 500-999
o 1,000-4,999
o 5,000-9,999
o 10,000 or more

<<demo_subcount>> How many subordinates do you have?

<<demo_industry>> What is your organization's primary industry?
o Biotech/Biomedical/Pharmaceutical
o Consulting and Business Services
o Chemicals
o Consumer goods
o Construction/Engineering
o Distributor
o Education
o Electronics
o E-marketplace (portals, auction, vert.)
o Energy
o Financial services/Banking
o Financial services/Insurance
o Financial services/Securities and Investments
o Financial services/Other
o Food/Beverage
o Government
o Healthcare/HMOs
o Hospitality/Travel
o IT Vendors
o Logistics/Transportation

o Manufacturing/Industrial (non-computer)
o Media/Entertainment
o Metals & Natural Resources
o Non-profit
o Retail/E-commerce
o Real Estate
o Telecommunications/ISPs
o Utilities
o Other (please specify) _____ <<demo_industry_more>>

<<demo_education>> What is your highest level of education?
o 8th Grade or lower
o 9th Grade
o 10th Grade
o 11th Grade
o 12th Grade, without diploma
o High school diploma, or equivalent (e.g.: GED)
o Some college but no degree
o Associate degree
o Bachelor's degree (e.g.: BA, AB, BS)
o Master's degree (e.g.: MA, MS, MEng, Med, MSW)
o Professional school degree (e.g.: MD, DDS, DVM)
o Doctoral degree (e.g.: PhD, EdD)

***[Only ask this question if <<demo_education>> >= 9th grade]***
<<demo_train_hs>>Have you received training through a high school course in programming?
o Yes
o No

***[Only ask this question if <<demo_education>> >= Some college but no degree]***
<<demo_train_univ>>Have you received training through a college or university course in programming?
o Yes
o No

***[Only ask this question if <<demo_education>> >= Some college but no degree]***
<<demo_major1>>What was your major or primary area of study in college?

***[Only ask this question if <<demo_education>> >= Master's degree]***
<<demo_major2>>What was your primary area of post-graduate study?

<<demo_train_job_prog>>Have you received on-the-job training in programming (for example, having another person explain to you how to program)?
o Yes
o No

<<demo_train_job_web>>Have you received on-the-job training in web page creation (for example, having another person explain to you how to create web pages)?
o Yes
o No

<<demo_train_job_ss>>Have you received on-the-job training in spreadsheets (for example, having another person explain to you how to create spreadsheets)?
o Yes
o No

<<demo_train_job_db>>Have you received on-the-job training in databases (for example, having another person explain to you how to create databases)?
o Yes
o No

<<demo_train_job_bi>>Have you received on-the-job training in business intelligence reporting (for example, having another person explain to you how to create business intelligence reports)?
o Yes
o No

<<demo_education_more>>How does your education and training compare or contrast with that of your subordinates?

<<demo_subscriber>>Are you an Information Week subscriber?
o Yes, I am a current subscriber
o I used to subscribe but do not now
o I have never subscribed

In order for us to contact you if you've won the grand prize of $500 or one of the five $100 checks to be awarded by CMP Media Inc. and your *free* report detailing the results of the study please be sure to enter your contact information below:

Name_____

Title: _____

Company_____

Phone_____

Email_____

Without compromising the confidentiality of your responses, would you be willing to speak to an *InformationWeek* editor for an upcoming story about these issues?

[] Yes

[] No

Thank you for participating in this survey!