A Methodology for Operational Profile Refinement

Sebastian Elbaum • UNL • Lincoln
Smita Narla • UNL • Lincoln

Keywords: software reliability, operational profile, clustering, software reliability engineering

SUMMARY & CONCLUSIONS

Numerous reliability models and testing practices are based on operational profiles. Typically, a single operational profile is used to represent the usage of a system with the assumption that a homogeneous customer base executes the system. However, if the customer base is heterogeneous, estimates computed on a single operational profile may be inaccurate. A single operational profile does not reflect the diverse customer patterns and it only "averages" the usage of the system, obscuring the real information about the operations probabilities. Decisions made on these estimates are likely to be biased and of limited usefulness.

This paper presents a refinement methodology for the generation of more accurate operational profiles that truly represent the diverse customer usage patterns. Clustering analysis supports the refinement methodology for identifying groups of customers with similar characteristics. Empirical stopping rules and validation procedures complete the refining methodology. A complete example of the methodology is presented on a large application. The example evidences the different perspective and accuracy that can be obtained through this refining methodology.

1. INTRODUCTION

The system's operations and their associated occurrence probabilities constitute an operational profile. Operational profiles provide quantitative descriptions of how the user will operate a software system (Ref. 1). In other words, they characterize system usage based, for example, on the average behavior exhibited by friendly customers on beta versions of the system. The quantified customer usage information given by the operational profiles, has been successfully employed to: (1) guide the testing effort on functions and operations that are important to the customer, (2) provide data for reliability estimations, and (4) perform a more appropriate allocation of resources.

In general, single operational profiles may accurately represent systems exhibiting a fairly consistent customer usage pattern. Consistent patterns are the product of a homogeneous customer usage base. However, when it comes to heterogeneous customer usages, it may be inaccurate to represent the system usage with a single operational profile. One operational profile will not reflect the true operational diversity of an entire system with varied usage patterns and configurations. On the contrary, a single operational profile would obscure the existence of groups with particular and different usage patterns, diminishing the opportunities of improvement based on accurate operational profiles.

As the characterization of system usage increases its role in most current practices of testing and reliability estimation, having accurate operational profiles becomes essential. In this paper we present a profile refining methodology that identifies significant groups of similar customers in order to generate more accurate operational profiles.

2. RELATED WORK

The importance of operational profiles has been widely investigated by many in the past. It is interesting to note that a large number of researchers have identified the need to generate more accurate operational profiles. The following paragraphs present a summary of some of the work that motivated our effort.

Woit (Ref. 2) described a technique for the specification of accurate operational profiles for modules. She emphasized the need of accurate operational profiles for operational – based statistical estimations, such as operational reliability. While Woit concentrated in developing accurate operational profiles for modules, Weyuker (Ref . 3) observed that reliability is not solely associated with the software, but really represents a user's view (operational profiles in our terminology), where different users with different operational distributions view the software and test suits differently. This observation supports our point that applications with heterogeneous customer bases are very difficult to represent with a single operational profile. Voas (Ref. 4) believes that apart from the telecom and the avionics industries, most other industries do not enjoy access to accurate operational profiles. Cheung (Ref. 8) also defined a user-oriented software reliability figure of merit to measure the reliability of the software system with respect to a user environment. Finally, Juhlin (Ref. 7) discussed the generation of accurate operational profiles by decomposing them into components to reduce broader variation in customer usage. She further emphasized the importance of operational profiles by

considering them the foundation of software reliability engineering.

3. PROBLEM DESCRIPTION

Several research efforts in the past indicate that the estimation of accurate operational profiles is always a difficult task (Ref. 5). One of the main reasons is the presence of large differences in the way customers use the system; that is, high variance in the customer base. The usage of a single operational profile for products with heterogeneous customer base may lead to inaccurate (sometimes biased) results, because no single operational profile can represent the entire usage pattern of a heterogeneous customer base. This lack of accuracy may result, for example, in poor reliability estimates.

As a simple example, let us consider a product with a single operational profile and a heterogeneous customer base to observe its impact more clearly. Let us suppose that there exists a set of users, *Set A*, who use a specific set of the products operations more frequently than some other set of users, *Set B*, and that the operations remain unexplored by a certain set of users, *Set C*, mainly because this particular set of users may not need to use those features. If there were some faults in these particular set of operations, they would go unnoticed by the users belonging to *Set C*. So the users of *Set C* may feel that the product is reliable as it effectively caters to their needs. Users of *Set B* may feel that the product is somewhat reliable as it meets all their needs "most" of the times. Users of *Set A* may have a complete different opinion of the system's reliability.

Using only a single operational profile to represent these users might be misleading. For example, if the customer base under consideration is dominated by the users of *Set A*, then the reliability estimates will be low even if users from *Set C* have high regards for the product. The opposite would happen if the users of *Set C* are majority. It is evident that the high variance among the customers may obscure or bias the reliability estimates based on operational profiles. One of the possible solutions for this problem is the decomposition of the single operational profile into three different ones, each one reflecting each set. Our methodology leads in that direction.

4. METHODOLOGY

An operational profile is normally generated based on the behavior exhibited by a set of friendly customers using a preliminary or previous version of the product or a similar product. The computation of the operational profile is based on the percentage of usage received by each operation $\boldsymbol{O}$ defined for the system, where the percentages are computed over the whole sample of customers. Then, the operational profile $\boldsymbol{op}$ can be described as a vector of operations probabilities such that,

$$op = [op_1, op_2, ..., op_i, ..., op_n] \quad \textbf{(1)}$$

where $\boldsymbol{n}$ is the number of operations defined for the system, and $\boldsymbol{op_i}$ is the probability of executing operation $\boldsymbol{i}$ such that,

$$op_i = \frac{\sum_{j=1}^{C} o_{ji}}{\sum_{j=1}^{C} o_{j*}} \quad \textbf{(2)}$$

where $\boldsymbol{C}$ is the total number of customer profiles, $\boldsymbol{O_{ji}}$ is the number of times operation $\boldsymbol{i}$ was executed by customer $\boldsymbol{j}$, and $\boldsymbol{O_{j*}}$ is the total number of operations executed by customer $\boldsymbol{j}$.

Our methodology is unique in that it attempts to repeatedly partition the collected system operational data so instead of ending with a single operational vector $\boldsymbol{op}$, a system will have a series of $\boldsymbol{op}$ to describe each group of customers more accurately. The fundamental idea behind this methodology is that the operational profile for a given cluster of customers, say $\boldsymbol{op_{clk}}$ (Equation set 3), would reflect more accurately the operation of the software by the cluster of customers $\boldsymbol{k}$, than a general single operational profile $\boldsymbol{op}$.

$$\begin{aligned} op_{cl1} &= [op_{11}, op_{21}, ..., op_{i1}, ..., op_{n1}] \\ op_{cl2} &= [op_{12}, op_{22}, ..., op_{i2}, ..., op_{n2}] \\ &... \\ op_{clk} &= [op_{1k}, op_{2k}, ..., op_{ik}, ..., op_{nk}] \\ &... \\ op_{clm} &= [op_{1m}, op_{2m}, ..., op_{im}, ..., op_{nm}] \end{aligned} \quad \textbf{(3)}$$

The partitioning process begins by grouping the individual customer usage patterns on the basis of their similarities. The grouping is generated employing a statistical technique called clustering analysis. Specific operational profiles are then generated for the identified groups following Equation 2.

Initially, the methodology considers all the collected individual customer usage data, so there are as many clusters as usage patterns. Then, the individual customers are sequentially analyzed to measure the distance among them. The distance between two members represents the similarities or dissimilarities between the customer usage data. Customers are placed in the same cluster if their usage patterns are similar. If the individual data are relatively far from each other, then the customers might become members of different clusters. The most similar customers are first grouped, and these initial groups are merged according to their similarities. The agglomerative process is repeated until the "desired" number of clusters is found. Figure 1 contains a summary of the refinement process.

The following sections present a more detailed description of this profile refinement technique. Section 4.1 constitutes a short introduction to clustering analysis. Section 4.2 expands on the choices made for the analysis. Section 4.3 presents the stopping criteria and validation procedure. The last section presents the methodology assumptions.

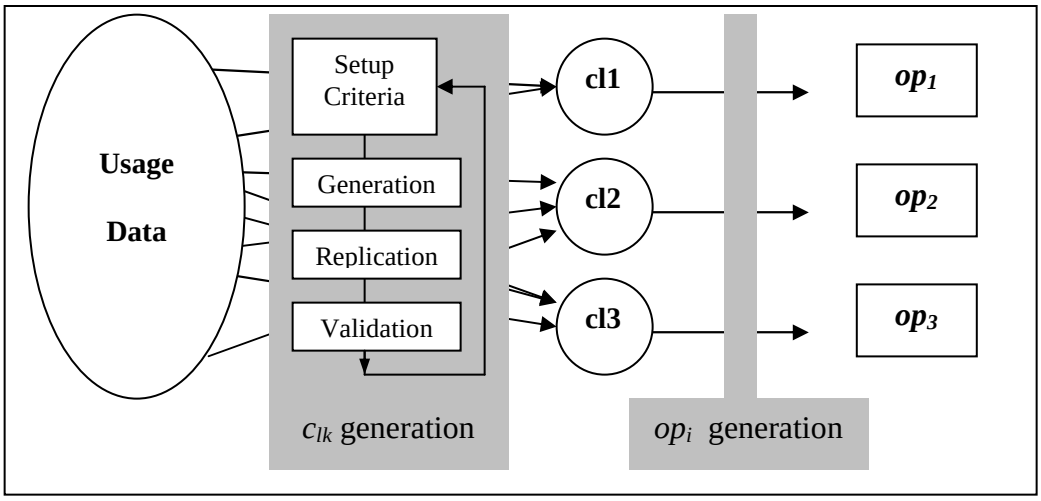

Figure 1. Profile Refinement Methodology

4.1. *Applying Cluster Analysis to Usage Data*

Cluster analysis[1] is a multivariate data analysis technique which partitions a population into clusters, each of whose elements are more similar to one another than to objects of other clusters. It assumes that each member of the population has a set of measurable attributes that constitute the inputs for clustering analysis and allow the comparison among the members of a population on the basis of similarities or distances.

The distances among the customer usage data are multidimensional because there is more than one operation per system. The simplest way of measuring distances between objects in a multi-dimensional space is to compute Euclidean distances[2]. This is the most commonly used distance within the clustering analysis arena because it is simple to compute, it uses raw data, and the distance between any two objects is not affected by the addition of new objects to the analysis (which may be outliers). The Euclidean distance is the geometric distance in the multidimensional space. It is computed as:

$$\text{Distance}(x,y) = \{\Sigma_i (x_i - y_i)^2\}^{1/2} \qquad (4)$$

At first, when each customer represents its own cluster, the distances between the customer usages are defined by the chosen distance measure. However, once several customers have been linked together, a linkage rule is needed to determine when two clusters are sufficiently similar to be linked together.

There are numerous possible linkage rules to consider. For example, the single linkage rule groups clusters by merging the nearest neighbors (the most similar customer). This linkage rule produces clusters chained together by only single objects that happen to be close together. Alternatively, we may use the neighbors across clusters that are furthest away from each other ensuring that all customers within a cluster are within a maximum distance from each other (complete linkage). We have chosen the Ward's method, which uses an analysis of variance approach to evaluate the distances between clusters. It assigns a given customer to the group that will receive the smallest increase in variance by including that customer. Ward's method is known to be effective in the generation of compact groups of well-distributed size clusters and is available in most statistical packages.

4.2. *Determining the Number of Meaningful Clusters*

Like many other multivariate techniques, cluster analysis presents the problem of how many clusters to keep. On one hand, the lower bound for the number of clusters to keep is one. If only one cluster is identified, the resulting ***op*** constitutes the same profile provided by the typical procedure, generating no refinements. One could conclude then, that the customer base is relatively homogeneous or that the partitioning stopping criteria is not restrictive enough. On the other hand, the upper bound for the number of clusters is the number of individual customers originally available to compute ***op***. This result may be due to an extremely heterogeneous customer base where every customer is intrinsically different, or very strict stopping criteria that may have resulted in too many operational profiles for them to be practically valuable.

Although some rules have been proposed (Ref. 10), there is no consensus on how to establish stopping criteria among statisticians. Hence, we have opted to define a set of criteria useful from the practitioner's perspective to apply this methodology. Under this methodology, we have defined three rules for cluster consideration:

[1] Most books on multivariate statistics include cluster analysis. As an example we cite Ref .9,10.

[2] If we had a two- or three-dimensional space this measure is the actual geometric distance between objects in the space (i.e., as if measured with a ruler).

- Each cluster needs to have a minimum of ***e*** customers for it to be worth representing in a separate op_{clk}, where ***e*** is determined by a cost/benefit analysis associated to having additional profiles. Setting ***e*** = 1 would nullify the impact of this rule by allowing the generation of clusters containing only one customer.
- The cluster structure needs to be validated through a number of replications. A replication is a repetition of the cluster analysis on a subset of the original customer data. A successful replication generates a cluster structure consistent with the initial one. The greater the number of successful replications, the more likely the cluster distribution is consistent. The variables to be defined are the number of replications, and the size of the randomly selected subsets. The number ***e*** needs to be adjusted based on the selected size of the subsets.
- The clusters reflect reasonable and interpretable structures. Although the methodology up to this point can be automated, high-level validation of the cluster structure is required for final corroboration. [3]

4.3. *Methodology Assumptions*

The correctness and completeness of the customer usage data constitutes the major assumption of this methodology. The usage data, as the input to the methodology, bounds the usability and applicability of the refinement. If the collection process does not capture the true nature of the customer usage patterns, then the refinement methodology will not be able to identify useful groups and it may even bias the profiles in the wrong direction.

A set of minor assumptions involves the use of clustering analysis. First, in the computation of the distances we have applied the Euclidian method, which uses raw data. If the measured attributes belong to different scales, the distances can be greatly affected and, consequently, the results of cluster analyses may be very different. Then, it is always necessary to ensure that the raw operation frequency measures are standardized or that a different distance computation approach be chosen. Second, most clustering methods do not consider sources of error and variation, making them very sensitive to outliers. As a result, the final configuration of clusters should always be carefully examined. This problem is reduced by the methodology through the replications. As the number of replications is increased, the cluster structure is re-validated for consistency.

Last, the methodology focuses on the refinement of operational profiles with minimal direct consideration of external factors. The cost/benefit analysis performed to compute ***e*** and the number of replications is left entirely for the practitioner. The methodology assumes the correctness of such parameters, as well as the correct interpretation of the cluster structure.

[3] In future work, discriminant analysis could be used to confirm the cluster structure, and to identify which operations are the main differentiators of groups.

5. STUDY

In order to demonstrate our methodology, we selected the widely used ***gcc***, the GNU(GNU's Not Unix) compiler almost de-facto under Unix environments. Gcc constitutes a very large system (in the order of millions lines of code) and it serves a heterogeneous customer base including a wide range of usage patterns (e.g., from novice students to professional programmers, from 8086 platforms to Risc based workstations). To serve all these environments, gcc provides a set of 312 flags that can be manipulated by the user. These flags constitute the available operations under gcc.

For our study, we selected 52 users (from a pool of over one hundred) representing a broad range of customers. Although this selection does not constitute a representative sample of the real usage diversity on gcc, it is enough to illustrate the possible refinement that can be obtained through the methodology. Table 1 contains a list of the subjects we selected. Observe that some of the subjects are represented by their product name (instead of their company or developer's name) to facilitate the explanation and posterior interpretation.

5.1. *Setup and Initial Clustering*

With the assistance of a commercially available statistical tool, clustering analysis was performed on the raw customer usage data. Only those operations that exhibit some variability among the sampled customers were included in the analysis (34 in total). The rest of the operations did not differentiate customers so they could not assist in the clustering. The minimum number of applications per cluster, ***e***, was set to 5 (approximately 10% of the sample size). In order to present the results of the analysis and facilitate its interpretation, we decided to utilize a tree diagram called a dendogram. In a dendogram, each branch represents a customer or a cluster of customers. The branches merge at a certain distance indicated by the x-axis, where smaller distances correspond to a larger similarity. For example, some customers are merged at no distance indicating that they are identical. Other clusters of customers merge further, indicating larger usage dissimilarities. Figure 2 includes the complete dendogram.

The analysis revealed two large clusters (visible in Figure 2), one including the students, and the other representing the non-student group, labeled as professional developers from now on. Within the student cluster, we can recognize two smaller clusters differentiating advanced from novice students (advanced student use additional facilities associated with debugging and more specialized libraries). Within these smaller clusters, some students differ slightly from others, but they are not numerous enough to constitute a new group. Within the professional cluster, we find three smaller groups. The biggest of the three is comprised of general user utilities. Language and library related applications belong to the third group. We could not find an

Number	Users/Apps	Short Description
1	Abuse	Source code used for game building
2	Barcode	Library/command-line for barcode generation
3	Cgicc	C++ class library for writing CGI applications
4	Cim	Simula compiler
5	Chess	Chess game
6	Emacs	Extensible editor and computing environment
7	Fileutils	Generic tools for file operations
8	g77	Cross compiler
9	Gawk	Free software version of 'awk'
10	Gcl (Clisp)	C and Lisp source files to build a common lisp system
11	Git	Tools for simple, daily file and system management tasks
12	Goose	Statistical computation library
13	Gperf	Hash function/table generator
14	Gprolog	Prolog compiler
15	Grep	Pattern matcher
16	Less	A paginator similar to "more" or "pg"
17	Libtool	Shared libs support
18	Mailman	Discussion list manager
19	Make	Build procedure assistant (dependencies,compilation,linkage)
20	Mig	Mach 3.0 interface generator
21	Ncurses	Library to display/update text on text-only terminals
22	Motti	Strategy game
23	Oleo	Spreadsheet
24	Parted	Disk partition manager
25	Patch	File update through diffs
26	Sed	A stream-oriented non-interactive text editor.
27	sh-utils	GNU portable shell tool
28	Tar	‘tar’ archives manager
29	Texinfo	Manuals and on-line docs generator
30	Textutils	Text utilities
31	Wget	Web retrieval
32	Which	File locator
33	Indent	C program beautification
34-42	st1-10	Novice students 1 through 10
44- 51	ad1-8	Advanced students 1 through 8

Table 1. Gcc Study

appropriate characterization for the second group, although it includes applications with similar type of interfaces.

It is interesting to note that increasing the ***e*** threshold would clearly reduce the number of resulting clusters and increase the number of customers included in each cluster. If ***e*** was set to 20, the results will still consistently produce a student and a professional cluster, limiting the refinement process.

In general a smaller ***e*** would provide further refinement possibilities. However, there is a point of diminishing return. For example, there are no refinement gains by setting to ***e*** less than 5. This constitutes a major tradeoff in the refinement methodology.

5.2. *Replication and Cluster Validation*

In order to validate the identified clusters, we performed two replications. Given the small number of customers, only two replications were considered necessary for the validation procedure to gives us confidence that the cluster structure was stable and consistent (a larger number of replications may have been required with a larger sample of customers). In each replication, the same analysis was performed on a subset of the original customers.

For the first replication we selected all the even numbered applications, for the second replication we considered all odd numbered applications. A more formal procedure could have been established (e.g., random assignment), but this should suffice for illustration purposes.

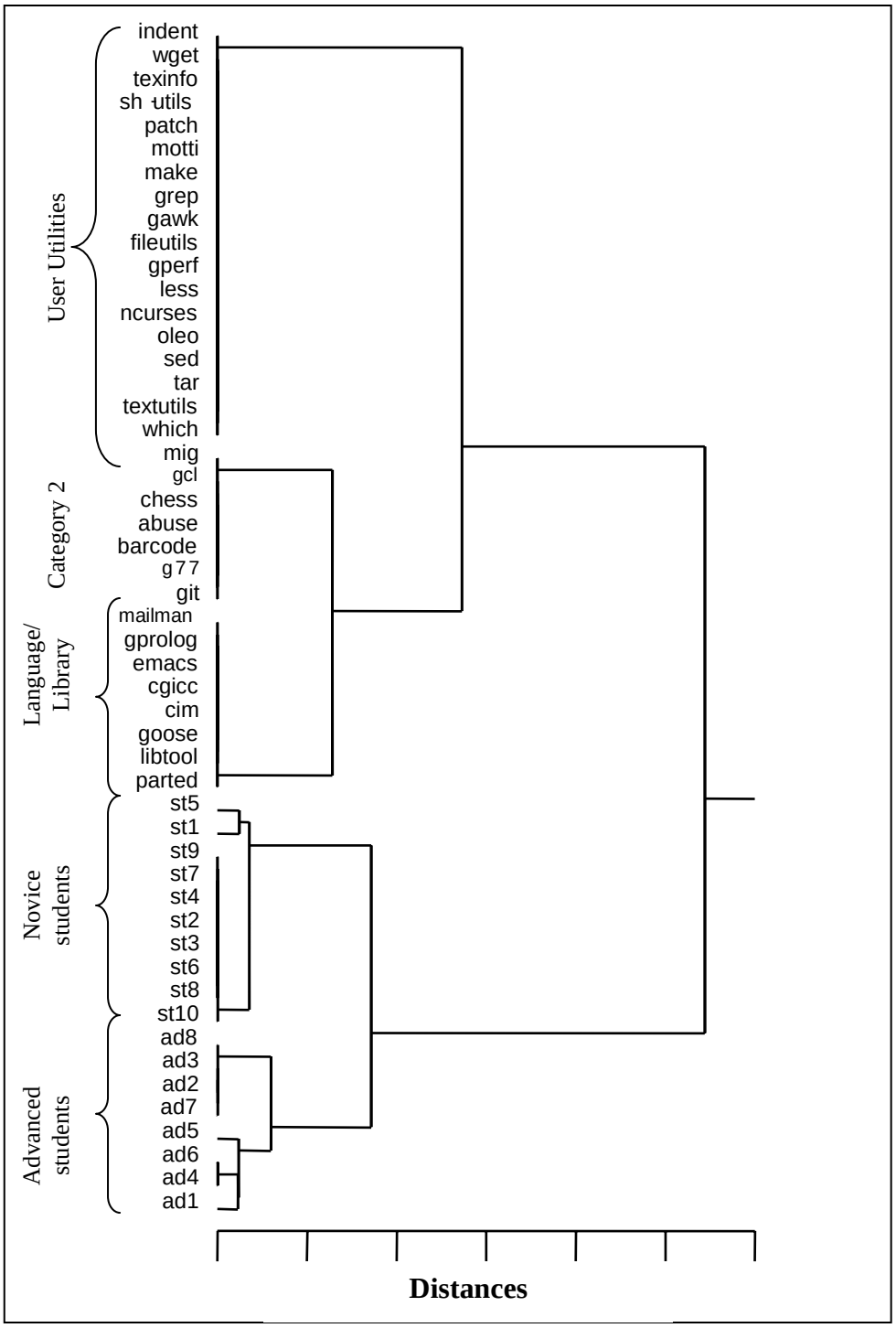

Figure 2. Dendogram for all data

The dendograms and corresponding analysis for both replications are presented in Figure 3.

First, the even numbered customers were considered (Figure 3 – left side). The two large clusters representing the students and the professional users are still clearly identifiable. Furthermore, the smaller groups within the clusters are consistent with the findings over the whole customer base. This illustration considering only the even cases reinforces the original cluster interpretation. The right dendogram in Figure 3 represents the second replication. The two large clusters are still present. The student clusters remained the same and were even more clearly defined because the chosen users did not show much disparity within each cluster (e.g., all novice students meet at distance zero). Within the professional cluster, most of the selected subjects fell into the large utilities groups. The other two smaller groups within the professional

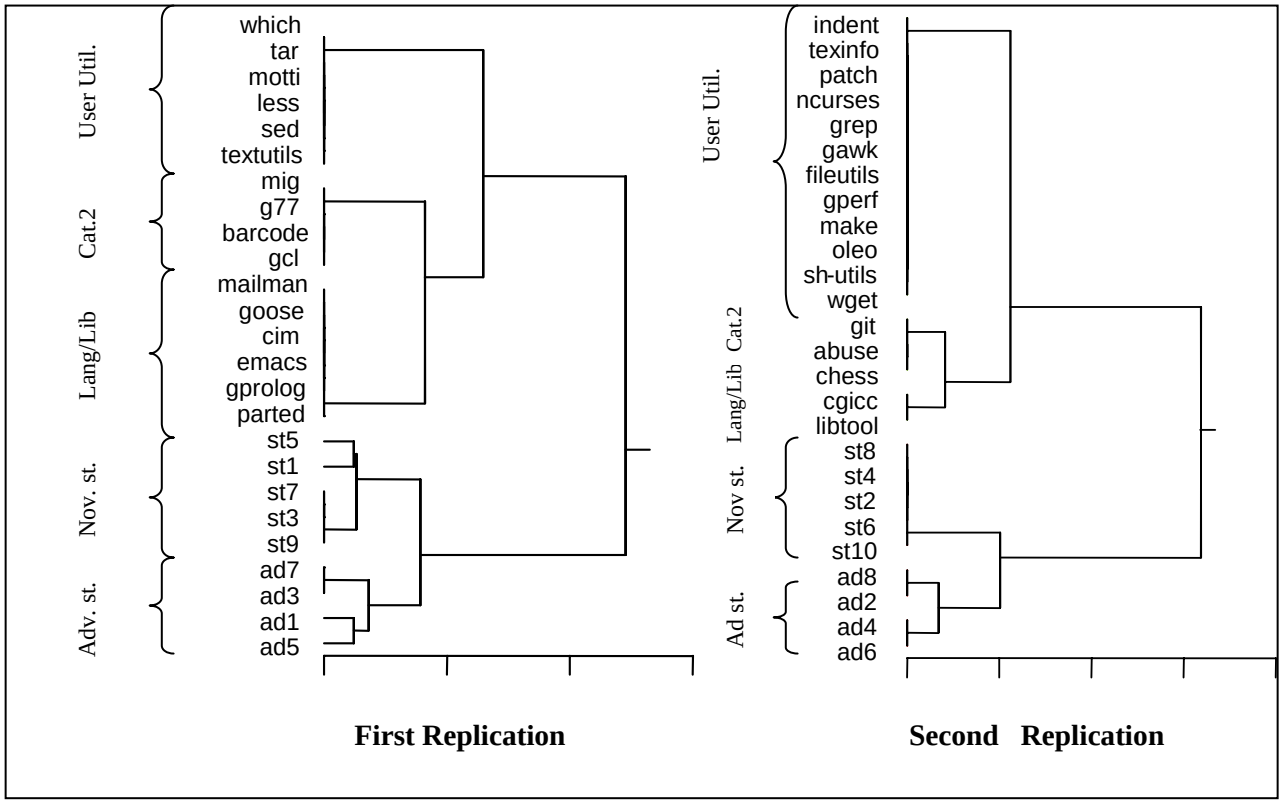

Figure 3. Clustering Validation

cluster denoted the only discrepancy. They remained consistent with the previous assessment but were too small to be considered independent. Hence, although the results were not exactly the same, they were fairly consistent with the original clustering and the first replication.

5.3. *Assessing the Refined Profiles*

In this section we illustrate the impact of the refinement of the original raw data in the resulting operation al profiles. Table 2 presents operational profiles at three refinement levels. The operation profile generated directly from the raw data (following the traditional computation) constitutes level zero. Level one incorporates the first level of refinement by considering the major two clusters of customers found in previous sections. Further refinement is presented at level two, which includes the smaller and more specific clusters. For illustration purposes, only 8 operations are included in this table.

Note that some operations probabilities changed

Refinement Levels			**Operations**							
			1	2	3	4	5	6	7	8
Zero - Traditional			0.19	0.13	0.19	0.14	0.03	0.09	0.03	0.07
One	Students		0.32	-	0.29	0.07	0.14	-	0.14	-
	Professionals		0.16	0.16	0.16	0.16	-	0.12	0.16	0.09
Two	Students	Novice	0.5	-	.45	-	-	-	-	-
		Advanced	0.22	-	0.19	0.11	0.22	-	0.22	-
	Professionals	Utilities	0.14	0.14	0.14	0.14	-	0.14	0.14	0.14
		Libraries	0.17	0.17	0.17	0.17	-	0.17	-	-
		Interfaces	0.2	0.2	0.2	0.2	-	-	-	-

Table 2. Refinement Assessment

considerably after the refinement process. For example, operation 1 is a very common operation with a 19% probability of being executed under the traditional operational profile. That probability estimate increases drastically if we only consider the customer segment represented by students (level 1 – students). It even increases to 50% if we only consider the novice students (level 2 – students – novice). However, the same operation has probability of 14% under the professional groups working on utilities (level 2 – professionals – utilities). Similar differences may be observed throughout the table. Table 2 just exemplifies how the refinement process produces more accurate operational profiles given a heterogeneous customer base.

6. FUTURE WORK

This research effort has shown the potential of the operation profile refining methodology. However, the methodology needs to be extended and further validated. The expansion of the methodology is a necessary step to account for the tradeoffs associated with the existence of additional operational profiles, and to facilitate the interpretation of the cluster structure. Conducting similar studies on systems with known groups of customers will allow additional measuring of the goodness of fit of the methodology.

ACKNOWLEDGEMENTS

This work was supported in part by a grant from NASA Nebraska Space Grant -EPSCOR.

REFERENCES

1. J. Musa, “Software reliability engineering”, 1998, McGrawHill.
2. D. Woit, “Specifying operational profiles for modules”, *Proceedings of the International Symposium on Software Testing and Analysis*, June 1993, pp 2-10.
3. E. Weyuker, “Using Failure cost information for testing and reliability Assessment”, *ACM Transactions on Software Engineering and Methodology*, Vol. 5, No. 2., April 1996, pp 87-98.
4. J. Voas, “Deriving of operational profiles for mass-marketed industry”, Technical Report at RST: www.rst.com/papers, 2000.
5. R. Horgan, A. Mathur, A. Pasquini, and V. Rego, “Perils of software reliability modeling”, *SERC-TR-160-P, Software Engineering Research Center*, Purdue University, February 1995.
6. K. Chruscielski and J. Tian, “An operational profile for the cartridge support software”, *Proceedings of the International Symposium on Software Reliability Engineering*, November 1997, pp 203-212.
7. B. Juhlin, "Applying software reliability engineering to international PBX testing", *Procedings of the International Conference on Testing Computer Software*, June 1992, pp 165-176.
8. R. Cheyung, “A user-oriented software reliability model”, *IEEE Transactions on Software Engineering*, Vol. 6, No.2, March 1980, pp 118-125
9. R.Johnson and D.Wichern, “Applied Multivariate Statistical Analysis”, Third Edition 1992, Prentice Hall.
10. B.Everitt and G.Dunn, “Applied Multivariate Data Analysis”, Edward Arnold, 1992.

BIOGRAPHIES

Sebastian Elbaum, PhD
107 Ferguson Hall
Computer Science and Engineering Department
University of Nebraska-Lincoln
Lincoln, NE 68688-0115, USA
Email: elbaum@cse.unl.edu

Dr. Sebastian Elbaum is an Assistant Professor at University of Nebraska-Lincoln, where he holds a J.D. Edwards Professorship. His research interests are software measurement, software testing, software reliability, and intrusion detection. He received his Ph.D. and MS at the University of Idaho. He has a Systems Engineering degree from Universidad Catolica de Cordoba, Argentina. He is a member of the IEEE, Computer Society, Reliability Society and ACM.

Smita Narla
Computer Science and Engineering Department
University of Nebraska-Lincoln
Lincoln, NE 68688-0115, USA
Email: snarla@cse.unl.edu

Smita Narla is a graduate student at University of Nebraska-Lincoln, where she is pursuing her MS in Computer Science. She has obtained her B.Technology in Computer Science and Engineering from Srivenkateswara University, India, in 1998.