World Applied Sciences Journal 21 (Mathematical Applications in Engineering): 50-55, 2013
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2013.21.mae.99918

Statistical Techniques to Serve Quality of Software

[1]Shihab A, Hameed, [2]Asad A. Aljumaily,
[3]Mohammed I. Ahmed and [1]Jamal I. Daoud

[1]Faculty of Engineering,
International Islamic University Malaysia, 50728 Kuala Lumpur, Malaysia
[2]Institute of Technology, Baghdad, Iraq
[3]Iraqi University, Baghdad, Iraq

Abstract: Within the rapid development of the recent computer-based applications; software quality becomes one of the critical issues to be considered. Software development is complex, costly, widely used and a core (heart) for critical and risky applications that have great affect on human life. Software testing is an essential issue in software quality. To get an efficient and effective testing we need a proper set of test data. One of the difficult and expensive parts of applying software-testing techniques is the generation of actual or meaningful test data since although there are several techniques and tools to automatically generate test data but mostly they generate meaningless data. This paper proposes a new model for generating meaningful test data. The model is based on using sample of real data and statistical techniques in generation process. The implementation results show that the samples of the generated data are meaningful and reflected the real world or field for testing processes. This leads to enhance the software quality.

Key words: Statistical Techniques · Software Quality · Test data · Meaningful data · Permutation

INTRODUCTION

In this century; information and knowledge become anew effective power for people, organizations and governments. Advancements in computer, communications and information technology leads to develop a more complicated and complex applications. Computer-based applications become the driving force for all aspects of life. The main challenge for information and communication (ICT) industry is to develop a qualified software product that meets business needs [1]. This variety needs makes software development a complex, costly and widely used as a core of many critical and risky applications that have serious affect on human life. Since 1990; the primary goal and challenge of software engineering shift toward producing high quality software and reducing the cost of the computer-based solutions [2, 3]. To check product quality; Software testing (SWT) is used as one of the essential tools to improve software quality. There are many SWT strategies and techniques. SWT is one of the complicated problems in software development life cycle, which is labor intensive, expensive and accounts for a significant ratio of software system development cost [4], around 40-50% of the total software development cost [5]. Software developments include set of risks that have different level of effect starting from unharmed to catastrophic. Software is now being applied in critical situations where some failure can be disastrous and therefore there is a real need to perform efficient and effective software testing to reduce the amount of risk of using software [6, 7]. There is big effort done to automate software testing process to reduce the cost of developing software. One of the most difficult and expensive parts of applying software testing techniques is the generation of test data. Test data could be prepared manually, copying from old version, or generated automatically by software tool. SWT requires sufficient, efficient and users' acceptable (meaningful) set of test data; this requires developing a test data generator such test data.

L.R shows several efforts done related to software test data generation [1, 8]. Test data can be generated either based on specification or code [9, 10, 11] or based on multi-agent [12]. The increasing number and

Corresponding Author: Shihab A. Hameed: Faculty of Engineering, International Islamic University Malaysia, P.O. Box 10, 50728 Kuala Lumpur, Malaysia.

complexity of SW products, make the manual preparation is unsuitable or inefficient because it's slow, costly and bias [13]. The secure information in the old version of data, the variety or difference in data description between the old copy and the under test copy and unavailability of sufficient volume of data in old version; these factors make the copying from old version is unsuitable or impossible [13, 14]. Automatic test data generation tools are required to generate the required volume of data in short time, less cost. Many of the current TDG tools are based on random number generation RNG technique. This RNG technique may suffers from less efficiency and in case of using set(s) of alphabets as resource data this could lead to generate a high rate of meaningless data. This leads to reduce user's confidence and trust in generated data and testing process as all. This leads to have a negative effect on the quality of software product. Since statistical methods are important tools to improve the quality of products and services [15]; this paper proposes such statistical methods and techniques to overcome such problem. This paper presents a model for generating samples of different meaningful data lists (different structures), based on permutation as generation techniques and sample(s) of actual data as resource data, to support SWT process.

Statistical Methods: Permutation: Statistic offers a great tool to serve all sciences. Statistic includes many techniques that can help in generating different sets of data objects from a limit set of objects using permutation or combination. Permutation is one of the widely used methods in statistics. "A permutation is an arrangement of all or part of a set of objects in some order" [16]. In this arrangement; some object is placed in the first position, another in the second position and so on, until all the objects have been placed. There are many permutation methods used in data generation process. Description of these permutation methods are explained in detail in [15-17]. These permutation methods are classified into (Selection with Replacement permutation, Selection without Replacement permutation, Selection from multiple groups of data, Partial replacement permutation and Combination). In this paper we will focus only on the first 3 methods of permutation, which include selection with replacement, selection without replacement and selection from multiple groups.

Selection with Replacement Permutation Method: Data generation, based on permutation with replacement method, is done by arranging the sample data elements according to all possible arrangements *with replacing* the selected element before next selection [16]. So that from a sample of **n** distinct elements; this technique can select at random **k** times, when the order of selection is important, a maximum of ($\mathbf{n} * \mathbf{n} * \mathbf{n} * \ldots * \mathbf{n}$) (for **k** times) with exactly ($\boldsymbol{n^k}$) permutations [17].

Selection Without Replacement Permutation Method: Data generation, based on permutation without replacement method, is done by arranging the sample data elements according to possibilities of arrangement *without replacing* the selected item before the next selection [15, 16]. So that from a sample of **n** distinct elements; this technique can select at random **k** times when the order of selection is important, a maximum of ($\mathbf{n! / (n-k)!}$) permutations [17]. The first distinct element is selected in **n** ways, the second distinct element in **n-1** ways, the third distinct element in **n-2** way and the k_{th} distinct element in **n-k+1** ways. The product is: $\mathbf{P(n, k) = n * (n-1) * (n-2) * \ldots\ldots * (n-k+1)}$.

Selection from Multiple Groups of Data Method: In permutation problems, some times, there are many kinds of data that can be classified into several groups such as n_1 of the first kind, n_2 of the second kind, ... and n_k of the k_{th} and if the sum of (n_1, n_2, n_3, ..., n_k) equal to n. Then number of all possible distinct permutations constructed from these groups is $\mathbf{(n!/(n_1! * n_2! * \ldots * n_k!))}$; all of equal probability of a random selection [16]. These generated permutations are constructed from elements belong to different groups and arranged in different orders. In case of using fixed order, such that the first element in all records of the list is selected from one group, the second element is selected from another group and so on, then the permutation of selecting one element from its group is $\mathbf{ni^1}$. According to multiplication theory; The permutation for selecting **k** elements from multiple groups are $\mathbf{P = (n1^1 * n2^1 * \ldots)}$

$\mathbf{= (n1 * n2 * n3 * \ldots).}$

Test Data Generation: Implementation with Permutation: Generation techniques based on permutation methods are suitable to generate similar-compound, not similar-compound, or composite lists from a given sample(s) of resource data to serve different users.

- Permutation with replacement technique is suitable to generate a list of similar compound records from one sample of resource data. The generated record holds a duplicated value in contiguous fields.

It is suitable to generate lists like full names (first name, middle name and last name) from list of single names, or list of student marks in four exams (1st exam mark, 2nd exam mark, 3rd exam mark and 4th exam mark) from list of marks.

- Permutation without replacement technique is also suitable to generate a list of similar compound records from one sample of resource data. It is a filtering to the previous one by removing the records with duplicated values. It is suitable to generate data like a list of full names under a restriction that the single name should not appear more than once in each record.
- Multiple-groups permutation technique is very useful to generate not-similar compound or composite lists of data from many samples of different types of data. It offers flexibility, to the users, in generating the different list(s) of data. Such as generating personal information list, birth information (country, state, date) list.

Although data generation techniques based on permutation methods offer a flexible and useful tool for generating lists of data, but there are some limitations or problems such as:

"The maximum volume of generated data according to the selected permutation method is less than, or greater than, required volume of data". This problem is solved by increasing or decreasing the size of sample data, select a more suitable permutation technique sine different techniques gives different volume of data from same sample(s), or duplicating, or eliminating, a previously generated list of data using sampling methods such as sequential, systematic, or random [18].

Main Modules and Components of Software Tdg Model: This test data generation model consists of the following main modules or components: Figure (1) shows the main components of the data generation model.

Setup Module: this module specifies the data item, list, to be generated, its structure, its specification, its components, fields, specifications and generation technique(s) used in generation.

Data Items Library Module: This library is constructed from collecting and unifying the data items and all it is related structures used by many applications of the same or similar goal. It contains all data items that can be generated by this model and all possible structures for each of these data items. As example full name, or date each have many structures used by different users.

Resource Data Model: contains the inserted sample of data for each data item to be used in the generation process.

Generation Engine Model: Contains the generation process and the generation routines based on permutation and sampling methods. These routines are used by generation engine to generate the required list(s) of data.

Implementation Steps for Software Tdg Model: In this section we present the steps for generation of different meaningful software test data lists according to the previous TDG model:

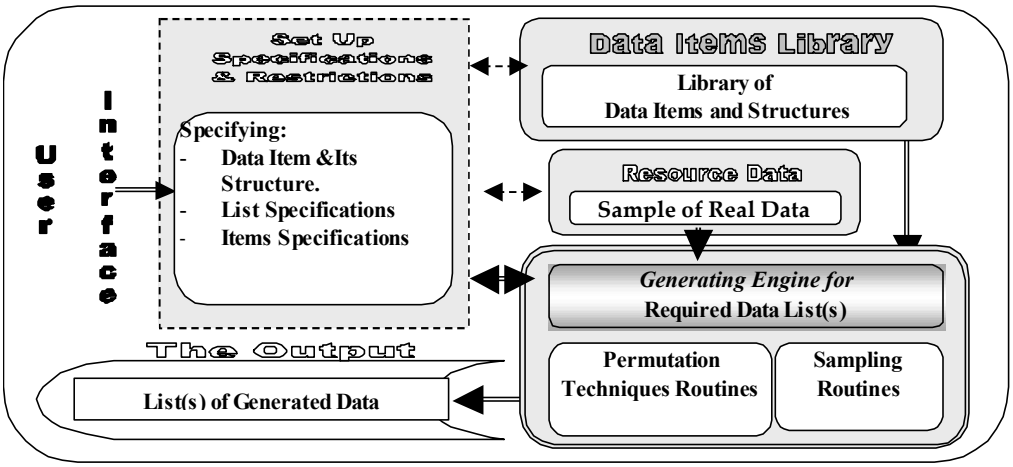

Fig. 1: Main Components of Data Generation model

Setup: In this step the user setup the requirements for generation by specifying the data item, list, specifications, its components and their specification. List specifications include name, type (simple or structure), its structure, required size to be generated and margin, main and secondary generation techniques and output device. Simple field specifications include name, type (integer, real, character, Boolean), [format], Lower & upper bounds and precision for the numeric type, Maximum length for the character type, resource of sample data is it a default or inserted by a user, the sample data size. Structured field specifications include name, type (similar compound, not similar compound and composite), size of sample data and resource of sample data (users / saved files). In case of generating not similar or a composite list we need to specify all fields that constructed the selected structure and all its related specifications.

Selecting the Sample Data: In this step; user inserts the sample(s) data directly, or select it from the previously stored default sub-library.

Validating: The sample(s) data is validated according to its related specifications and limitations such as type, format, lower and upper limits for numeric type and maximum string length for character data type. The validated data then used as a resource data for the generation process.

Generation: In this step, the generation engine calls a selected generation routine(s), such as permutation or sampling technique and the validated sample of data to generate the required list(s) of data according to the generation strategy if it is one or multiple phase generation.

TDG Implementation Results: To generate a list of meaningful data; we feed the model with a sample(s) of actual data and selecting the suitable structure and permutation technique(s). In this implementation we use a very small sample(s) just to show the way of generation. To generate huge volume of data we can use big size sample(s). Generating lists of meaningful data based on different permutations is shown as follow.

To generate a list of full names under permutation with replacement, we need to:

- Sample of single names such as (Ahmed, Isa, Aziz, Taha, Hadi){5 names}.
- Structure of the full name item such as: (First-Name Mid-Name Last-Name)
- Permutation with replacement Generation Technique.

Table (1) shows the generated list of full names according to the above limitations, where N= sample size, K= length of generated data (here is 3), p =$(5)^3$ =125 full names. This list contains many of duplicated values in the full name such as (Ahmed Ahmed Ahmed) or (Hadi Hadi Aziz) and so on.

- To generate a list of full names with no duplicated names; we use the permutation without replacement as generation technique. The generated list is shown in table (3). Where P = 5! / (5-3)! =60.
  The result shown in table (2) shows no duplication of the same name for the person, his father and his grandfather. This result is more realistic than the previous one (Table 1) since we find such duplication.

Table 1: List of names generated by permutation with replacement

	Ahmed Ahmed Ahmed	Ahmed Ahmed Isa	Ahmed Ahmed Aziz	Ahmed Ahmed Taha	Ahmed Ahmed Hadi
	Ahmed Isa Ahmed	Ahmed Isa Isa	Ahmed Isa Aziz	Ahmed Isa Taha	Ahmed Isa Hadi
	Ahmed Hadi Ahmed	Ahmed Hadi Isa	Ahmed Hadi Aziz	Ahmed Hadi Taha	Ahmed Hadi Hadi
	Isa Ahmed Ahmed	Isa Ahmed Isa	Isa Ahmed Aziz	Isa Ahmed Taha	Isa Ahmed Hadi
	:	:	:	:	:
	Isa Hadi Ahmed	Isa Hadi Ibrahim	Isa Hadi Aziz	Isa Hadi Taha	Isa Hadi Hadi
	↓	↓	↓	↓	↓
	Hadi Ahmed Ahmed	Hadi Ahmed Isa	Hadi Ahmed Aziz	Hadi Ahmed Taha	Hadi Ahmed Hadi
	:	:	:	:	:
	Hadi Hadi Ahmed	Hadi Hadi Isa	Hadi Hadi Aziz	Hadi Hadi Taha	Hadi Hadi Hadi

Table 2: List of names generated by permutation without replacement

Ahmed Isa Aziz	Ahmed Isa Taha	Ahmed Isa Hadi
Ahmed Aziz Isa	Ahmed Aziz Taha	Ahmed Aziz Hadi
:	:	:
Ahmed Hadi Isa	Ahmed Hadi Aziz	Ahmed Hadi Taha
Isa Ahmed Aziz	Isa Ahmed Taha	Isa Ahmed Hadi
:	:	:
Isa Hadi Ahmed	Isa Hadi Aziz	Isa Hadi Taha
↓	↓	↓
Hadi Ahmed Isa	Hadi Ahmed aziz	Hadi Ahmed Taha
:	:	:
Hadi Taha Ahmed	Hadi Taha Isa	Hadi Taha Aziz

Table 3: List of Dates generated by permutation for multiple group

1 Feb 95	2 Feb 95	4 Feb 95	5 Feb 95	1 Jun 95	2 Jun 95	4 Jun 95	5 Jun 95
1 Oct 95	2 Oct 95	4 Oct 95	5 Oct 95	1 Feb 97	2 Feb 97	4 Feb 97	5 Feb 97
1 Jun 97	2 Jun 97	4 Jun 97	5 Jun 97	1 Oct 97	2 Oct 97	4 Oct 97	5 Oct 97
1 Feb 98	2 Feb 98	4 Feb 98	5 Feb 98	1 Jun 98	2 Jun 98	4 Jun 98	5 Jun 98
1 Oct 98	2 Oct 98	4 Oct 98	5 Oct 98	1 Feb 99	2 Feb 99	4 Feb 99	5 Feb 99
1 Jun 99	2 Jun 99	4 Jun 99	5 Jun 99	1 Oct 99	2 Oct 99	4 Oct 99	5 Oct 99

Table 4: New list of Dates generated by permutation for multiple group

1/02/1995	2/02/1995	4/02/1995	5/02/1995	1/10/1995	2/10/1995	4/10/1995	5/10/1995
1/02/1997	2/02/1997	4/02/1997	5/02/1997	1/10/1997	2/10/1997	4/10/1997	5/10/1997
1/02/1998	2/02/1998	4/02/1998	5/02/1998	1/10/1998	2/10/1998	4/10/1998	5/10/1998
1/02/1999	2/02/1999	4/02/1999	5/02/1999	1/10/1999	2/10/1999	4/10/1999	5/10/1999
1/02/2000	2/02/2000	4/02/2000	5/02/2000	1/10/2000	2/10/2000	4/10/2000	5/10/2000

- To generate a not similar compound list such as dates (date of birth, date of appointment, etc); we need to three samples for data representing the day, month and years as input beside using the permutation for multiple groups as generation technique. Based on the following selection of structure, input samples and permutation technique:

  - The selected structure for date is: (Day (N) Month (Ch) Year (NN))
  - Input samples are (1, 2, 4, 5) for days, (Feb, Jun, Oct) for months and (95, 97, 98, 99) for years.
  - Generation technique is multiple group permutation.

Since the generated items, records, are of fixed order, so the generated list is represented in table (3); where $P = 4 * 3 * 4 = 48$.

We may use different structure or format for date such as (Day (N) Month (NN) Year (NNNN)) then we use different samples as input.

We may use different structure and sample input to generate a different set of date. The following example shows the change in structure and input data then we find different result.

- The selected structure for date is: (Day (N)/ Month (NN) /Year (NNNN))
- Input samples are (1, 2, 4, 5) for days, (02, 10) for months and (1995, 1997, 1998, 1999, 2000) for years.
- Generation technique is multiple group permutation.

Since the generated items are of fixed order, so the generated list is represented in Table (4);

Where $P = 4 * 2 * 5 = 40$.

DISCUSSION AND CONCLUSION

Software quality is the main target for any software engineer. Software applications become a driving force for human life in almost all aspects. Software testing is essential way to get qualified software product.

This paper works toward solving one of the critical problems in software testing by introducing a model for generating meaningful test data using sample of real data and set of permutation methods. The model is capable of generating any required volume of meaningful test data, from small sample of real data, by customizing different permutation techniques. Test data generation TDG is very helpful especially in stress or boundary testing. The generated data are mostly meaningful data, which reflects the specification, domain and environment of the population; this will increase confidence and trust in generated test data and the testing process as whole. Other advantages of using this method are low cost, less effort and time and better usage of the computing resources. This model considers the users viewpoint in generating required test data. All these advantages lead to have successful software and consequently serving quality of software.

REFERENCES

1. Hitech Tahbildar and Bichitra Kalita, 2011. Automated software test data generation: direction of research, International journal of computer science & engineering survey (IJCSES), 2(1):.
2. Mynatt, B.T., 1990. Software Engineering with Student Project Guidance. US, Prentice-Hall International Editions.
3. Pressman, R.S., 1997. Software Engineering: A Practitioner's Approach. (4th ed.). Singapore, McGraw-Hill Book Company.
4. Ferguson, R.C. and B. Korel, 1996. The Chaining Approach for Software Test Data Generation. ACM Transactions on Software Engineering and Methodology, 5(1): 63-86.
5. Offutt, A.J., Jin Zhenyi and Jie Pan, 1999. The Dynamic Domain Reduction Approach to Test Data Generation. Software Practice and Experience, to appear in 1999.
6. Offutt, A.J., 1991. An Integrated Automatic Test Data Generation System. Journal of Systems Integration, 1(3): 391-409.
7. Hameed, S.A., A. Deraman and A. Hamdan, 1998. A Framework for Database Test Data Generator, Technical Report FTSM / MEI LT- 48, University Kebangsaan Malaysia.
8. Shahid Mahmood, 2007. A Systematic Review of Automated Test Data Generation Techniques, Master Thesis Software Engineering Thesis no: MSE-2007: 26 October 2007.
9. Gautam Kumar Saha, 2008. Understanding Software Testing Concepts, ACM Ubiquity 9(6): 12-18.
10. Prasanna, M., S.N. Sivanandam, R. Venkatesan and R. Sundarrajan, 2005. A survey on automatic test case generation, Academic Open Internet Journal, www.acadjournal.com, 15.
11. Mark Utting, Alexander Pretschner and Bruno Legeard, 2006. A Taxonomy of Model-based testing, April 2006.
12. Siwen Yu and Jun Ai, 2010. Software Test Data Generation Based On Multi-Agent, International Journal of Software Engineering and Its Applications, 4(1).
13. Hameed, S.A., A. Deraman and A. Hamdan, 2000. A Framework for Generating meaningful test data, Information technology symposium-Y2K, 11-12 April, University Kebangsaan Malaysia.
14. Hameed, S.A., 2000. Meaningful Test Data Generation based on Statistical Methods, Ph.D. Thesis,Faculty of information science and Technology, University Kebangsaan Malaysia.
15. Douglas C. Montgomery and George C. Runger, 1994. Applied Statistics and probability for engineers, John Wiley and Sons Inc.,
16. Ronald E. Walpole and Raymond H. Myers, 1993. probability and statistics for engineers and scientists, 5th ed., MacMillan Publishing Company.
17. Richard W. Hamming, 1991. The art of probability for scientists and engineers, Addison- Wesley Publishing Company.
18. Lawrence L. Lapin, 1975. Statistics meaning and methods, Harcourt brace Jovanovich Inc.,