Operational Profiles for Statistical Testing of Distribution Management System

ILIJA BASICEVIC[1]
ILIJA KUPRESANIN[2]
MIROSLAV POPOVIC[1]

[1]University of Novi Sad
Faculty of Technical Sciences
21000 Novi Sad- Serbia
[2]JP SRBIJAGAS
Narodnog Fronta 12
21000 Novi Sad - Serbia
[1]ilibas@uns.ac.rs,miroslav.popovic@rt-rk.com
[2]ilija.kupresanin@srbijagas.com

Abstract. Each generation of software systems is becoming more complex. Also, software is becoming more important because today critical infrastructure systems depend on software. This paper presents the method applied in testing of a complex software system. For complex systems, it is very important to measure their reliability. Statistical testing based on operational profiles is a de facto industrial standard for this purpose. As a case study, we used Windows-based distribution management system that is used in electric power distribution utilities. A library that contains the analytical functions subsystem was tested. The paper gives an overview of the system and module being tested, and of the statistical method we applied. The main contribution of this paper is development of operational profiles for a real-world complex system. Two of the operational profiles we have developed for testing of the system are presented in detail. Another contribution is a new approach which supports generation of test cases on-the-fly: during execution of implementation under test on a test bed. This is possible by joining together the test case generator and the test bed.

Keywords: power distribution management system, statistical testing, operation profiles, software testing, software reliability.

(Received December 8th, 2010 / Accepted June 16th, 2011)

1 Introduction

Statistical testing is a technique that originates from quality assurance in automatic production of mechanical and electrical devices. Given the importance that complex software systems have in everyday life, sometimes in critical applications, it is no surprise that this technique has found its place in software engineering, too.

Along the time scale, software of today is getting more complex. The statistical testing using operational profiles is a method for measuring reliability of complex software systems. An example of critical infrastructure system that depends on software is Distribution Management System (DMS) in electrical power distribution network, the case study that is used in this paper. The complexity of this class of software systems stems from the complexity of mathematical models of distribution networks, which are the basis for the software model. As an example of complexity, the input space for DMS in the case of the State of Texas measures 13 million of variables [8].

This paper presents operational profiles we have developed for DMS system. In our approach, the test case generator and test bed are joined together. This way, system supports generation of test cases during execution of implementation under test on the test bed.

The DMS Software is a software package for performing technical tasks in electric power distribution utilities. This software tool enables utility's staff to design and manage the network's power and automation equipment in order to maximize the quality and quantity of the electrical energy supplied to the consumers.

The DMS software is modularly organized package with three-tier software architecture. It is based on standard software solutions that should allow for simple integration with other standard software and hardware equipment found in the environment of electricity distribution (SCADA Systems, equipment for medium voltage (MV) automation, etc.).

The general DMS software architecture is briefly described as follows. The first tier is a relational data base. The middle tier integrates static technical and historical data with dynamic data (available for example from third party SCADA systems). In the third tier, there are user clients - Microsoft Windows applications. In the actual software architecture, some of third tier clients represent a shell for the DMS analytical functions system.

The most important applications of the third tier are [2]:

- Front-end application for data base editing. Used for editing of parameters of network elements, of their connectivity and finally their graphic representation in the form of a network diagram.

- Multiple-view user interface for visualization of supply and distribution substations and MV and LV network, as well as for managing and monitoring distribution network state. It contains integrated DMS analytical function system.

- SCADA system user interfaces.

2 Related Work

Whittaker and Thomason described a method for statistical testing based on Markov chain model of software usage in [14]. In this fundamental paper, authors discuss the construction of Markov chain and show how analytical results associated with Markov chains can aid in test planning. An innovation in this method is that test sequences generated and applied to the software are used to create the second Markov chain to encapsulate the history of the test. The paper also presents a stopping criterion for testing process. However, the example Markov chain for a hypothetical graphical user interface (GUI) is a very simple one. In the conclusions and prospects for future work authors mention that they are investigating more abstract models. To that end, we are still missing more complex and abstract domain specific models in the available literature. That is exactly the place where our paper tries to provide contribution by dealing with the construction of real-world operational profiles for the complex library of power distribution functions. The parameters of application programming interface (API) functions of this library are complex data structures rather than simple keystrokes. In contrast to [14], the stopping criterion in our paper is simply reaching the given target reliability figure.

In their essay on application of statistical science to testing and evaluating software intensive systems [5], Poore and Trammell present a high-level overview of the statistical testing process. In the section on building usage models they provide valuable rules and hints, e.g., how to expand states and arcs and how to create sub-models within a model, but they do not cover sensitive issue of how to generate inputs. Generating inputs gets really complicated when they are not simple types, such as strings, etc. For example, in the case of the DMS library the input may be the part, or complete model of the power distribution network, which is essentially a graph whose nodes and arcs are complex data types. In our paper, we provide some guidance and experience how to generate such inputs. Authors of [5] also present analytical method of assigning probabilities to state transitions within usage model. Alternatively, in our paper, we practice direct assignment of probabilities based on domain specific knowledge of power distribution systems, because they provide rather accurate representation of the reality. But, it is worth mentioning that analytic assignment has its place in practice when the probabilities are unknown.

Guen, Marie, and Thelin have proposed coverage measures for both states and transitions of the usage model and an approach to estimate the reliability from Markov chains by using their tool MaTeLo in [3]. After necessary definitions they introduce a heuristic for the construction of: equivalence classes, estimators in spaces of internal states and input vectors, and global indicators for individual transitions and for the whole chain. Then they present limitations of solutions proposed by Whittaker and Sayre, and propose their own method which combines the Sayre's solution and calculation based on the equivalence classes. In contrast to

[3], we do not measure model coverage directly. We use an alternative approach by selecting the test case length based on the domain-specific knowledge and then use significance quality indicators (SLi)s as the measure of generated test cases quality. The stopping criterion in this paper is that the mean significance confidence level is above 20% and that the target reliability figure is reached. Another difference between [3] and this paper is that we use the reliability estimation method proposed by Woit [15], which is based on the purely statistical hypothesis approach and therefore does not require construction of equivalence classes.

In another paper [4], closely related to [3], Guen and Thelin report on their practical experiences with statistical usage testing by means of their tool MaTeLo. The company's Alitec experience shows that the model construction takes between 0.5 and 4.5 days per KLOC (1000 lines of code). They have found that even for small software products, the size of the model would be very large, and have commented that it is probably the major drawback of that sort of modelling. In addition, they reported that time to finish testing was between 1.5 and 8.3 days/KLOC for 5 example projects, but they do not provide data about the target reliability and coverage. In our own experience, the most consuming part of modelling was gathering domain-specific knowledge of the behaviour of the power distribution network, e.g. knowledge about how switches and tap changers are operated. That activity lasted several months, and it is still work in progress. Once the basic knowledge was collected, we were able to construct various operational profiles within a working week each, so that time for a model per MLOC (million lines of code) became extremely low, e.g. 5/2x106 = 2.5 days/MLOC. Typical testing campaign in our experience lasts for 2-3 weeks, yielding time to finish test in the range of 7-11 days/MLOC.

MaTeLo [4] contains a usage model editor. Based on entered usage model, the back end test case generator generates automatically TTCN-3 test cases. Similarly, in [10], [9], [7], there is usage model editor based on GME. Depending on entered usage model, textual test cases are generated and executed in a JUnit based test bed. In both approaches, there are three separated steps:

1. Input of the model
2. Generation of test case
3. Execution of test case in test bed

Also in the prototype tool presented in [12], the test case generator and the test oracle are separated. In their approach, test case generator is Java framework used for testing Java classes. In this paper, we present an approach where steps 2 and 3 are joined. The first step is realized by writing C++ code.

The flat and hierarchical models supported by current statistical testing environments, such as MaTeLo [3], tend to become enormous very quickly as systems grow in complexity. This fact is recognized as major drawback in [4]. In his study [13], Weber proposes a solution to this issue. The proposed solution is to use parallel models because their usage provides exponential reduction in model size. Essentially, Weber creates an operational profile of a system by extending its requirement specification model. The approach is demonstrated with the example of Flight Guidance System (FGS) provided by Rockwell Collins. Although [13] presents an interesting approach, it requires a specification model in language such as RMSL-e (based on Requirements State Machine Language, RSML) as its input which may not be readily available in case of legacy systems and development of such models would be very costly. For example, the library we are dealing with in our paper contains millions of lines of FORTRAN code. Additionally, this legacy code is sequential by its very nature. Therefore, reducing the size of operational profile based on introduced parallelism is not possible.

Most of research on operational profiles is focused on operations and little is said about operation parameters. Shukla et al. [12] offer a solution to this issue by introducing support for defining constraints on individual parameters, relationships between different input parameters (of the same or different operation calls), and between output parameters of calls and input parameters of subsequent calls. Unfortunately, they do not report in details what kinds of constraints and relationships are supported and how. Although their work seems promising, they demonstrated it only on a three rather small examples, namely Stack, Symbol Table and Forest (of abstract syntax trees) with 35, 128, and 234 lines of code, respectively. In our paper, we deal with the complex legacy library of power distribution functions. We provide parameter constraints and relationships by translating them into C++ code that directly manipulates power distribution network model used by the library.

3 Implementation under Test

This DMS analytical function system is a component that contains the domain specific knowledge of DMS software. It implements comprehensive set of sophisticated algorithms that enable efficient design, optimal operation and decision making referring to the whole equipment installed in the distribution network [6]. It realizes all technical tasks in distribution utilities in the following four modes of operation:

- Operation Management,
- Operation Planning,
- Development Planning,
- Simulation, Analysis and Training.

All analytical functions are developed on the basis of algorithms specially aimed for distribution networks, which enable performing both analysis and optimization of operation and development of very large radial and weakly meshed distribution networks.

The DMS analytical function system is implemented as a dynamic link library, named dmsapp.dll. This paper describes the method we have applied in the black-box testing of the library. The library is written in the FORTRAN programming language (version 10, as of today it is being ported to version 11). It is developed by development team, and tested by quality assurance team. It is an important module of the DMS system and realizes calculation of load flow, fault calculation, thermal monitoring and several other functions. Application Programming Interface (API) of the library contains functions for creation and destruction of the network model and invocation of specific calculations. There is a C language API for use in C applications. API functions will not be described here in detail. All calculations that the library and its API provide to developers are based on the software model of distribution network and this model is implemented in the library.

The software model of the distribution network in DMS system is a complex data structure. As has already been stated, it is the graph whose nodes and arcs are complex data types. Elements in the model are linked by identifications: for each element, in the data structure that represents it, there are fields for identifications of adjacent elements. In the general case, each network element is presented with three objects, containing different types of data. Those three objects comprise an abstraction hierarchy. The catalogue object contains catalogue data. Those are technical and administrative data that describe the class of objects. The set of similar objects after a period of use are assigned type data that is the result of statistical processing of measured data about the set. The basic data describe the specific element in case, and among others, contain the dynamic values. Thus, the catalogue object is the highest one in abstraction hierarchy and the basic data object is the lowest.

4 Testing Procedure

Testing of the library is realized using test environment we have developed. It is based on CPPUnit [1] framework for automated testing. In the setUp() method of CPPUnit test case, the test environment is prepared for the test. The tested library is initialized and the network model is loaded into the library. In the main test case method, which is registered to the CPPUnit framework, the finite state machine of Markov chain is executed. This is done by using OP State Machine class which is configured with particular states and their transitions. Each state transition implies invocation of functions that belong to the application programming interface of the tested library. In the tearDown() method, the test environment is closed down by freeing the used memory and by closing the handles of used system objects. It is possible that CPPUnit has defects, but during the testing we did not encounter them. If those defects manifested themselves, it would not make a problem for the testing process, because in that case, the asserts would fail.

The applied methodology is based on statistical testing methods described in [15], which are often used today, for example in Cleanroom Engineering [11]. This method implies that the functional correctness of DMS System is tested using operational profiles. The method is sufficient for testing provided that the operational profile correctly models the statistics of the usage of the tested product. An operational profile is modelled as a finite state machine. It can be represented as a graph, consisting of a certain number of states which are nodes of the graph. We can describe the state as a general condition of the software module. The edges are state transitions which are triggered by events that occur with certain probabilities. An event is either an external invocation of a member function of the tested module or a change of value of an externally accessible variable.

We have developed realistic, non trivial operational profiles for the DMS system. The API that is used for program invocation of DLL functions is rather complex because data structures that are used for data passing - as input and output arguments of function calls - are large and complex. This implies a large number of test cases required for testing of the analytical functions library.

The primary task of the testing tool is to generate requested number of test cases with the requested number of test steps. A test step represents an event that is defined in operation profile. The length of the test case represents the number of test steps in the test case. The more test cases are executed successfully, the greater is the system reliability. An important characteristic of our approach is that tests are generated and executed dynamically. Test cases are generated (based on specified operational profile) during execution of the system in the test bed. This feature is realized by joining together the test case generator and the test bed.

In the traditional statistical testing, the first step is to generate (based on the operational profile) test cases and to save generated test case in a file. In the next step, the file is read and the test case is executed in the test bed. On the other hand, in our approach, the two steps are joined. First, the test bed generates the next test step by selecting one of the state transitions available in the currently active state - according to probabilities of state transitions and then, generated test step is executed. This is repeated until the last test step in the test case is executed. Operational reliability of the software module is the probability that module execution, selected at random according to given operational profile, will not fail.

Reliability = 1 - (failure rate).
Module execution is considered to be the sequence of events issued to the module beginning with module initialization (or re-initialization) and ending with module termination (or re-initialization). Operational profile is a description of the distribution of input events that is expected to occur in module operation. System reliability is determined by formula:

$M = r^N$

where:
N - number of successfully executed tests,
r - reliability,
M - confidence
For example, for the system's reliability (with the confidence of 0.7%) to be 99%, 500 test cases ought to be executed successfully, for the reliability of 99.9%, 5000 test cases ought to be executed successfully, for the reliability of 99.99%, 50000 test cases ought to be executed successfully, etc. Confidence is the probability that module has reliability less than r and still passes N tests. Significance level represents the quality of test sample of operation profile. Significance level is calculated using next formulas:

$e_{\mathrm{ij}} = P_{\mathrm{ij}} * \sum f_{\mathrm{ij}}$,

$D_{\mathrm{i}} = \sum(f_{\mathrm{ij}} - e_{\mathrm{ij}}) * \frac{f_{\mathrm{ij}} - e_{\mathrm{ij}}}{e_{\mathrm{ij}}}$,

where:
P_{ij} - probability of event j in state i,
f_{ij} - real occurrence of event j in state i,
e_{ij} - expected occurrence of event j in state i,
D_{i} - discrepancy of state i.
For significance level calculation, it is necessary to form a table in which rows represent states and columns represent events. Elements of the table are real and expected occurrence of events (eij and Pij). Significance level is determined in χ-table for every state based on calculated discrepancy. The mean value of the significance level is calculated next. It is given as the arithmetical mean value of significance levels of all states. If the mean value of significance level is greater than 20%, the given sample of operation profile is valid.

The testing method also includes a method for estimating software reliability based on statistical hypothesis testing. The result is reliability estimation accompanied by the measure of confidence. The overall procedure comprises of the following steps:

1. Specification of an operational profile
2. Generation of random test cases based on the operational profile
3. Execution of test cases
4. Verification of test cases
5. Reliability estimation based on the model and the results of the verification of test cases

Testing process is modelled using Binomial distribution because it satisfies the following criteria:

- Testing is performed with replacement
- Tests are selected at random and they are mutually independent
- Test has only two possible outcomes (success, failure)
- The probability of failure does not change during the testing

In this testing, three operation profiles have been identified and implemented:

1. Operational profile for changing dynamic data of distribution scheme. This profile randomly chooses a number of elements of each type and randomly changes dynamic values of selected elements. As each switchgear element contains purpose field, the elements are classified according to the value of that field. The set contains 5% of all switchgear elements with the purpose field indicating supply line or reactor, 10% of those with purpose indicating three types of high voltage transformers or a feeder, etc. After each change, load flow is calculated once again. The validity of Kirchhoff laws is checked before and after series of changes of dynamic values of different elements.

2. Operational profile for scheme mutation. This profile is different from profile 1 because instead of dynamic values, the static structure of the network is changed. There are two basic types of mutation: scheme extension and scheme reduction. Extension is achieved by adding elements or groups of elements, and reduction by deleting elements or groups of elements from the scheme.

3. "Realistic" operational profiles. We have developed six operational profiles based on operations that network operators routinely undertake in their everyday work. Those are:

- Change of the position of tap changers,
- Activation of capacitor batteries and its impact on currents and voltages,
- Load growth simulation and monitoring of changes of currents in the transformer bay,
- Detection location and insulation of faults,
- Feeder looping,
- Load sharing between the two feeders (Feeder to feeder load shedding).

We describe here a typical test case for operational profile 1. First, the tested library is initialized and an existing network scheme is loaded. This is a scheme that has already been used with the DMS system and thus, it is assumed to be correct. This starting statement is necessary because in further steps, the network scheme would be modified.

The test procedure modifies the scheme and checks if the library would successfully process or recognize as invalid the subsequent states of the scheme. The scheme is stored in a set of binary files and placed in the common folder. It is loaded from binary files to the library by first placing it into an array of binary type data structures. If this step goes well, the scheme is loaded from the array into dmsapp.dll.

Using one of the described profiles, the loaded network scheme is modified (by modifying dynamic values of network elements), see 1. By invoking the load flow calculation or by checking the validity of Kirchhoff laws, test case verifies if the DMS system can process the modified network. The expected behaviour is that the library should either correctly perform the requested calculations or return an error code in the case of an invalid state of the input network. Any other behaviour (e.g., unsuccessful calculation, throwing an exception during calculation) is considered to be test failure.

For the operational profile 2 (scheme mutation), the scheme is modified (by modifying the network structure) before it is loaded into dmsapp.dll.

From the programmer's point of view, test execution is achieved by instantiating an object of class OPStateMachine that models the finite state machine of module execution. First, objects that represent specific states of operation profile are instantiated and their state transitions (including probabilities) are configured. Thus a vector of state objects is formed. This vector is a parameter of the OPStateMachine constructor. During the construction of the object, the state machine is executed. By applying probabilities that are given in the state machine, the test case is generated and executed dynamically. Next the significance level of the test is calculated. This calculation is based on two matrices: of expected and achieved occurrence of events. Based on the calculated value, it is decided whether generated test case is a representative sample.

5 Operation Profile for Changing Dynamic Data

The operation profile for changing dynamic data has the following three states: S_0 - Initial state, S_1 - Network loaded, ready for check procedures, S_2 - Check procedure done, network ready for the next modification. and seven state transitions which are represented with events that trigger them and state transition probabilities we used in testing:

1. LoadDMIAndSetLibOptions (100%),
2. CheckKirchhoffsLaws (100%),
3. ChangeCapacitorChangers (15%),
4. ChangeFuseStatus (20%),
5. ChangeSwgStatus (35%),
6. ChangeTRTapChangers (15%),
7. SetDynValueGenerator (15%).

In the following text, each of the events is explained. The LoadDMIAndSetLibOptions event triggers scheme loading. The following options are set for dmsapp.dll: Load Flow, Estimation, Performance and Switching. All Switching options are activated except grounded and energized and different levels.

The CheckKirchhoffsLaws event triggers the procedure in which the validity of Kirchhoffs Laws for the network is checked. The ChangeCapacitorChangers event starts the procedure in which the list of all capacitors with variable capacity is compiled, and then the capacity of each capacitor in the list is changed to a random value in acceptable interval. This interval is read in the catalogue object of the capacitor.

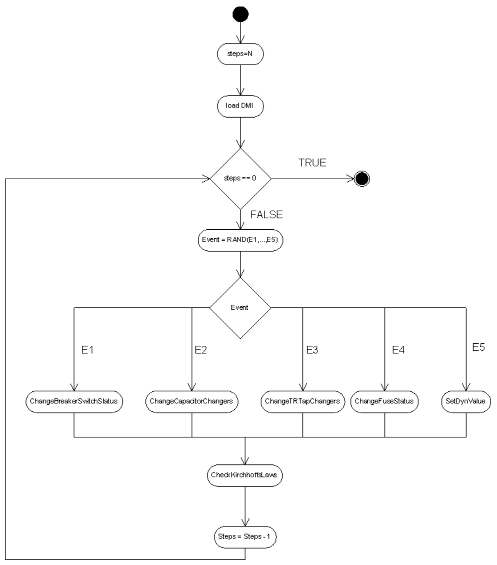

Figure 1: The UML activity diagram for the profile 1 (changing dynamic data)

The ChangeFuseStatus event starts the procedure in which the list of all fuses in the network is compiled. 10% of elements in the list are chosen at random and turned off, the rest are turned on. Before each turn on/off operation, the viability of the operation is checked, and upon each successful state change, it is checked whether the load flow calculation of the new network topology is possible. If not, the test is finished at that point.

The ChangeSwgStatus event starts the procedure in which the list of all switchgear elements in the network is compiled. Next, the elements in the list are classified by the value in their purpose field. For each purpose type, a set of randomly chosen elements A is formed. Set B, is subset of A and elements of B are chosen at random from A. The status of elements in B is set to open, and the status of all elements in A that are not in B is set to close.

The ChangeTRTapChangers event starts the procedure in which the list of all tap changers for high voltage transformers in the network is compiled. Next, values of elements in the list are randomly set in ranges that are read from their catalogue objects. This procedure is repeated for middle voltage transformers, too.

The SetDynValueGenerator event starts the procedure in which the list of all generators in the network is compiled. The regulation type of the generator can be:

1. Regulation of active power,
2. Regulation of active power and voltage,
3. Regulation of active and reactive power,
4. Regulation of active and reactive power and voltage.

Based on the regulation type, the values of elements in the list are changed:

1. Active power in the range (P_{min}, S_{nom}), where P_{min} is the minimal power of the generator and S_{nom} is the nominal power of the generator,
2. Reactive power in the range (Q_{min}, Q_{max}), where Q_{min} is the minimal reactive power and Q_{max} is the maximal reactive power
3. Voltage in the range (0.5*V_{nom}, 1.5*V_{nom}), where V_{nom} is the nominal voltage.

P_{min}, S_{nom}, Q_{min}, Q_{max}, and V_{nom} values are retrieved from the catalogue object for the generator. The probabilities of events are determined in the following manner. LoadDMIAndSetLibOptions and CheckKirchhoffsLaws have 100% probabilities to occur. The events ChangeTRTapChangers, ChangeCapacitorChangers and SetDynValueGenerator are assumed to be less probable to cause errors, so they have smaller probabilities. The events ChangeSwgStatus and ChangeFuseStatus are changing the topology of the network. Therefore, it is assumed that they can cause more errors and thus have greater probabilities.

Figure 2) presents the operational profile for changing dynamic data. There are three states. Upon loading the scheme, the system transitions from S_0 to S_1, and afterwards, upon checking Kirchhoff laws, it transitions from S_1 to S_2. It returns from S_2 to S_1 by changing dynamic values of certain network elements.

6 Feeder-to-Feeder Load Shedding

This operational profile is based on one of the scenarios of the realistic use of the DMS system. In the following paragraph the reader can find explanation of the testing procedure in this operational profile. The first step is to compile a list of all candidates. The candidate has to meet the following conditions:

- Switchgear element SWG is in off state
- SWG is contained in feeder bay TSM (transformer station medium voltage) or joint
- Bay should be connected to an energized bar

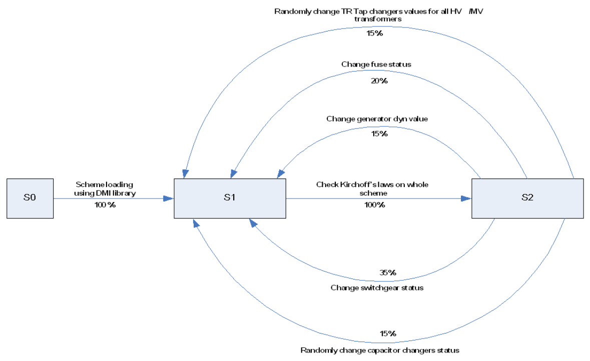
Randomly change TR Tap changers values for all HV /MV transformers
15%
Change fuse status
20%
Change generator dyn value
15%
S0
Scheme loading using DMI library
100 %
S1
Check Kirchoff's laws on whole scheme
100%
S2
35%
Change switchgear status
15%
Randomly change capacitor changers status

Figure 2: The operational profile for changing dynamic data

- SWG should be connected to two different feeders, belonging to different TSH (transformer station high voltage) elements

In the second step, an element from the list is selected at random (SWG_1) and the testing is started. Measured values for both feeders are recorded. The following variables are set: the feeder with greater load is F_{max}; TSH that contains F_{max} is TSH_{max}. Next, the direct path from TSH_{max} to SWG_1 over F_{max} is searched. A list of elements containing sections and TSMs in the direct path from TSHmax to the selected TSM is compiled. A list of all SWG elements that belong both to that path and to the feeder is compiled. If there are no SWG elements in the path that meet the requirements, the test is considered to be successful, and the execution returns to the beginning of step 2.

Next, SWG_1 is switched off. If it is not possible to execute load flow calculation, SWG_1 is switched to its former state, the test is considered to be successful, and the execution returns to the step 2.

From the list of SWG elements that are in the path and are switched on, one is selected (SWG_2). SWG_2 is switched off. If it is not possible to execute load flow calculation, SWG_1 and SWG_2 are switched back to their previous states, the test is considered to be successful, and the execution returns to the step 2.

Check of the validity of Kirchhoffs laws for the new state of the network is conducted. The new measured values for selected feeders are compared with old, and if the following conditions are met, the test is considered to be successful:

$Feeder1Load < Feeder2Load :$
$Feeder1Load < Feeder1LoadNew$
$Feeder2LoadNew < Feeder2Load$
$Feeder1Load > Feeder2Load :$
$Feeder1Load > Feeder1LoadNew$
$Feeder2LoadNew > Feeder2Load$

Afterwards, SWG_1 and SWG_2 are switched back to their previous states and the check of the validity of Kirchhoffs laws for the new state of the network is conducted. The execution returns to the beginning of the step 2.

The test is performed in the following manner. First, from the list of SWG_1, one by one, each of the elements in the list is selected. Next, the path from TSH to TSM containing SWG_1 is searched (over the feeder with greater load). Afterwards, all SWG elements in the path are added to the SWG_2 list and a SWG is selected at random from the SWG_2 list. In the last step, feeder to feeder load shedding is performed for the selected combination.

7 Conclusion

DMS is a large and complex software package for performing technical tasks in electric power distribution utilities. As already stated, DMS software systems are getting more complex every day. One important software metric is its reliability, and therefore, statistical testing by using operational profiles as a de facto standard for measuring software reliability is of special interest in development of DMS systems.

Two important contributions of this paper are the set of developed operational profiles for statistical testing of DMS systems and on-the-fly testing approach, which is achieved by joining together the test case generator and the test bed. In earlier test beds that are described in the available literature, the two were separated.

The operational profiles have been developed for testing of one module of distribution management system software - that is the module that contains analytical functions subsystem. Several operational profiles have been developed, two of which are presented here in detail (operational profile for changing of dynamic data, and feeder-to-feeder load shedding).

Besides providing us with a measure of reliability, the testing with the use of operational profiles has allowed us to identify certain software faults earlier in the development process.

8 Acknowledgement

This work was partially supported by the Ministry of Education and Science of the Republic of Serbia under the project No. 44009 and 32031, year 2011.

Authors thank Pavle Kuzevski, Jelena Djurica, Tereza Kovac, Snezana Crnogorac Jovanovic, Ranka Slijepcevic, Rodoljub Radivojevic and Ljiljana Dragojlic for implementation of operation profiles, and valuable discussions and feedback during realization of this paper.

References

[1] Cppunit - c++ port of junit. sourceforge.net/projects/cppunit.

[2] Dms software, windows for distribution networks. DMS Group, 2006.

[3] Guen, H. L., Marie, R., and Thelin, T. Reliability estimation for statistical usage testing using markov chains. In *International Symposium on Software Reliability Engineering (ISSRE)*, 2004.

[4] Guen, H. L. and Thelin, T. Practical experiences with statistical usage testing. In *Annual International Workshop on Software Technology and Engineering Practice (STEP)*, 2004.

[5] Poore, J. H. and Trammell, C. J. *Statistics, Testing, and Defense Acquisition*, chapter Application of Statistical Science to Testing and Evaluating Software Intensive Systems. National Academy Press, 1998.

[6] Popovic, D., Bekut, D., and Treskanica, V. *Specijalizovani DMS algoritmi*. DMS Group, 2004.

[7] Popovic, M., Basicevic, I., Velikic, I., and Tatic, J. Practical experiences with statistical usage testing. In *Annual Conference on Engineering of Computer Based Systems (ECBS)*, pages 377–386, 2006.

[8] Popovic, M., Basicevic, I., and Vrtunski, V. Practical experiences with statistical usage testing. In *Annual Conference on Engineering of Computer Based Systems (ECBS)*, 2009.

[9] Popovic, M. and Kovacevic, J. A statistical approach to model-based robustness testing. In *Annual Conference on Engineering of Computer Based Systems (ECBS)*, pages 485–494, 2007.

[10] Popovic, M. and Velikic, I. A generic model-based test case generator. In *Annual Conference on Engineering of Computer Based Systems (ECBS)*, pages 221–228, 2005.

[11] Prowell, S. J., Trammell, C. J., Linger, R. C., and Poore, J. H. *Cleanroom Software Engineering: Technology and Process*. Addison-Wesley Professional, 1999.

[12] Shukla, R., Strooper, P. A., and Carrington, D. A. Tool support for statistical testing of software components. In *Asia-Pacific Software Engineering Conference (APSEC)*, 2005.

[13] Weber, R. J. Statistical software testing with parallel modeling: A case study. In *International Symposium on Software Reliability Engineering (ISSRE)*, 2004.

[14] Whittaker, J. A. and Thomason, M. G. A markov chain for statistical software testing. *IEEE Transactions on Software Engineering*, 20(10), October 1994.

[15] Woit, D. M. *Operational Profile Specification, Test Case Generator, and Reliability Estimation for Modules*. PhD thesis, Queen's University Kingston, Ontario, Canada, 1994.