AUTOMATIC SIMULATION CALIBRATION: AN EVALUATION OF SEARCH-BASED METAHEURISTICS APPLIED TO IEC 61131-3 BASED AUTOMATION IN THE POWER SYSTEMS DOMAIN

Cláudio Silva, Rosaldo J. F. Rossetti, Jorge C. Pereira
Faculty of Engineering of University of Porto
Oporto, Portugal
E-mail: claudio.silva@efacec.com, rossetti@fe.up.pt, jpereira@inescporto.pt

KEYWORDS

Simulation, Automatic Calibration, IEC 61131-3, Power Systems Automation.

ABSTRACT

Industrial automation rely on PLC to perform real-time critical tasks. These tasks are commonly implemented using IEC 61131-3 and tend to be very complex due to current evolution of microprocessor capabilities. This complexity causes testing to be considered a very difficult task and the scope to be reduced to minimize effort. The application of automatic calibration technics could allow the identification of optimal or near optimal solutions for the set of inputs to allow a better test coverage and simulation accuracy.
This paper reviews current automatic simulation calibration methodologies and provides a description of an experiment of the application of search-based metaheuristics to IEC 61131-3 implemented algorithms in the context of power systems automation.

INTRODUCTION

Industrial automation in general and substation automation in particular, rely on programmable logic controllers (PLC) to perform real-time, critical tasks. These include fault detection, condition monitoring or energy metering just to name a few examples. The use of PLC in substation automation dates back to the end to the 1970s [Brand et al. 2003] and provided a control relay functionality programmed in Relay Ladder Logic [Wilson 1999]. Nowadays, almost all vendor specific approaches have taken advantage of the programming languages defined IEC 61131-3 [International Electrotechnical Commission 2003], making this standard the leading standard for defining behavior for these devices [John and Tiegelkamp 2010]. In addition, with the evolution of microprocessors technologies, these devices have become more flexible and capable thus allowing for a more complex definition of behavior. Because of this evolution, testing these devices is considered a very complex task. To reduce such complexity, engineers tend to reduce the scope to be tested, focusing only on a subset of behaviors that are deemed as relevant and representative of the full implementation.
More recently, we also see a growing trend for the use of simulation environments to allow a more comprehensive and thorough analysis of the behavior implemented in PLCs. These simulation environments allow for offline validation, removing the need for an actual physical device, and, for parallel execution of validation methodologies [Pereira et al. 2011; Carlsson et al. 2012]. However, the complexity of the behavior is still there… The use of automatic calibration technics for such simulation environments could allow to identify an optimal (or near optimal) set of input variables that would be able to minimize the workload when testing the implementations. At the same time, it would allow to gain more confidence on the developed solutions due to the increased test coverage and simulation accuracy.
This document starts by providing a literature review of simulation calibration techniques already applied in different contexts. It continues by describing an experiment designed to evaluate some of these techniques within the power system domain and its relevant results. It finishes by drawing conclusions from the results obtained and enumerating some enhancements that could be implemented to improve the base experiment solution.
To allow for a clear understanding of the information discussed in this paper, the following concept should be clear. This paper refers to automatic simulation calibration as the identification of variables, and corresponding sets of values, that have an impact on the execution of a simulation run of a specific model. It does not refer to the adjustment of simulation engine parameters to match the behavior of a real-world device. For the purpose of this paper, it is assumed that the simulation engine behaves as similar as possible to the physical system.

LITERATURE REVIEW

The following chapters provide a review of the methodologies used in automatic calibration of simulation. The review was not limited to the power system automation domain but focused on a more general approach giving preference to the target domain.
When trying to facilitate the calibration of simulation environments, two distinct approaches can be taken. The first would be to calculate exogenous variables values. This would be an application domain specific solution, because it requires an analysis and identification of patterns of external inputs. It requires access to real data, acquired by sensors or outputs of the system that would need to be processed to evaluate their applicability and evolution. Nevertheless, this would also generate more “real-world” test case scenarios. On the other hand, and since we are discussing a programming language based model, we could also work form the inside out. Analyzing the implemented behavior of the developed code, one can infer on the possible input variables that would have an impact on system behavior.

Since this is dependent only on the code to be executed, it is application domain independent, but will most probably also required a more extensive test suite to validate the entire model.

Exogenous Variables Calculation

In this first methodology, exogenous variables calculation, we attempt to calibrate input variables using real-world data. This data needs to be processed and two different approaches are applicable for this. Some authors propose the use of extensive data mining, determining the patterns within the collected data. Several authors, more prominently within the traffic simulation domain, propose the use of multiple information sources and sensors to calibrate simulation models using data mining. These include the combination of traffic flow sensor and geographical information into a data mining infrastructure to identify patterns for system behavior [Liu et al. 2011] or the use of clustering techniques to create flow-occupancy diagrams and thus allowing the determination of several characteristics such as critical occupancy, pre-queue flow [Kianfar and Edara 2013], vehicle speed [Jiang et al. 2012] or average daily traffic [Gecchele et al. 2011]. Others also propose the use of a Fisher Information Matrix to examine sensor data measurements and how their quality relates to the quality of the simulation model [Li et al. 2013] to allow for the creation of an automatic differentiation method to improve the data selection for use in model calibrations. Nevertheless, other application domains also have works targeting simulation calibration. The use of Rough Set Theory to identify dominant factors for urban growth and land-use changes [Wang et al. 2011] or the introduction of generic data mining techniques for calibration of building energy models [New et al. 2012].

Data mining would seem as the "best alternative" to calibrate a simulation because it uses real-world information. However, due to the need of an extensive data collection and adequate quality insurance mechanisms, this may be impractical to execute. To minimize the need for this extensive data collection, other authors propose the use of statistical methods to infer the possible values, or evolution of values, of each input variable. Within this approach, there are two algorithms commonly present in references: the Kalman Filter and the Markov Chain Monte Carlo.

The Kalman filter is an iterative process, separated in two steps, that infers parameter values based on observations. As examples of the application of this algorithm, Huang et al. introduce the use of a Kalman Filter to validate and adjust the simulated phasor values of a power generator [Huang et al. 2009] while Kalsi uses the same method to calibrate the inputs for an exciter and power system stabilizer model [Kalsi 2012].

Markov Chain Monte Carlo (MCMC) is an algorithm that allows sampling from a probability distribution taking into consideration the last sample that was taken to consider the next sample. Higdon et al. use MCMC to combine real world data with simulated data to calibrate the inputs of two example models, a charged particle accelerator and a spot-welding process [Higdon et al. 2004] while Papaefthymiou and Klockl uses MCMC to define the behavior of a wind turbine power generation [Papaefthymiou and Klockl 2008].

While these statistical methods require less real-world information, they still require an extensive analysis of the information. They require a well-known probability distribution to be determined to be able to extrapolate data from it. Similarly to data mining approaches, it requires that "enough", and of quality, data is available.

Model Analisys

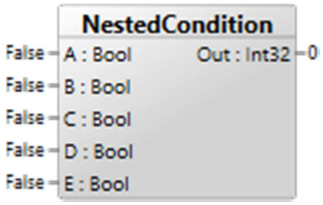
In this second methodology, the world outside no longer is of relevance. Only the model is. This type of calibration requires that the internal of the model is available and well known, thus allowing for a white-box analysis. This is similar to a software developer writing unit tests for his own developed code.

This type of approach is very common in software development communities [Finley et al. 1998; Zan Yang et al. 2001; McMinn 2004; Park et al. 2008; Jee et al. 2009; Wang et al. 2012; Braione et al. 2013; Wu and Fan 2014] and aims at generating a set of test cases that can comply with a determined objective. This objective can be ensuring a minimal code coverage, a minimal path coverage or the generation of a range of output values, just to name a few examples. This has been an area of study for almost 40 years [King 1976; Miller and Spooner 1976] and have produced different approaches. Authors have compared these approaches [Braione et al. 2013] and defined three categories to describe them. The first, (1) random testing generates test cases based on random input values. (2) Dynamic symbol execution assumes an initial input state and evaluates code execution with those inputs, trims the executed branches and evaluates a new set of input values that will transverse the other branches. Finally, search-based testing, views the problem as an optimization problem aiming at maximizing a coverage objective. More specific approaches to languages defined in IEC 61131-3, Functional Block Diagram (FBD) in this case, have also been proposed such as creating a formal definition for FBDs data flow [Jee et al. 2009; Wu and Fan 2014]. This allows a better evaluation of program definitions, especially the data flow path, and thus providing means to provide a better code coverage.

EXPERIMENTAL SETUP

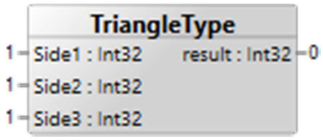
Within the evaluation scope of this paper, and due to the limited access to real-world data, the focus was the above described model analysis. It was used as base the IEC 61131-3 simulation engine developed by Efacec [Ferreira et al. 2008; Lemos et al. 2011]. This platform allows the definition of behavior in two of the languages defined in the international standard, Structured Text and Functional Block Diagram. It includes the required parsers, compilers and linkers to generate binary files to implement the behavior on target devices but also a simulation engine that can execute those binary files within the toolset.

To be able to use this platform and capture the necessary outputs, it was required to introduce some modifications to the base toolset, not present in the commercial version. It was created an instruction execution counter to allow verification of code coverage and execution paths. This was implemented ensuring that minimal impact was created for the execution of test runs.




```
IF A THEN
  Out := 1;
END_IF;
IF A AND B THEN
  Out := 2;
END_IF;
IF A AND B AND C THEN
  Out := 3;
END_IF;
IF A AND B AND C AND D THEN
  Out := 4;
END_IF;
IF A AND B AND C AND D AND E THEN
  Out := 5;
END_IF;
```

Figure1: Nested Condition Algorithm Overview




```
IF Side1 <= 0 OR Side2 <= 0 OR Side3 <= 0
THEN
  result := 4;
ELSE
  IF Side1 = Side2 AND Side2 = Side3
  THEN
    result := 1;
  ELSE
    IF Side1 = Side2 OR
       Side2 = Side3 OR
       Side1 = Side3
    THEN
      result := 2;
    ELSE
      IF Side1 <> Side2 AND
         Side2 <> Side2 AND
         Side3 <> Side1
      THEN
        result := 3;
      END_IF;
    END_IF;
  END_IF;
END_IF;
```

Figure 2: Triangle Type Identification Algorithm Overview

Testing Algorithms

For testing the capabilities of the metaheuristics, three algorithms were selected. Two were generic algorithms, commonly used as benchmark for many testing papers: nested conditions and triangle identification. The other is an interlocking condition algorithm used in substation automation solutions developed by Efacec. An interlock condition algorithm is used within this domain to evaluate, based on a view of the state of the system, if a specific operation is allowed or not. For example, a circuit breaker may not be allowed to close when an adjacent breaker is already closed or when the device that is acquiring the breaker position is indicating an error on the readings. Since it represents proprietary information, it is not fully reproduced here. The first two algorithms are described in 1 and 2 while the last is depicted in 3 with only the input and output variables. It is relevant to note that while the two benchmark algorithm are defined using a very small amount of input variables, with a small set of possible value combinations, the interlock algorithm uses more than 40 different values, expanding the possible values combination sets. This is one of the motivations that exist to facilitate in the identification of the relevant input sets and attest the statement made previously that testing this type of behavior is a complex task.

Figure 3: Interlock Algorithm Overview

Metaheuristics Algorithms

The metaheuristics algorithms implemented were the following three. A random test case generator, a tabu search and an evolutionary algorithm based test case generator. The random provides test sets by generating arbitrary values for each of the input variables. The tabu search is adapted from the original definition were elements that provide an undesired state are tabued for a determined time. Instead, each element that is selected to be modified in an iteration is tabued in the following iterations. This was done to steer the algorithm to generate different states by choosing different input variables. The evolutionary algorithm choses variables to change in each iteration based on a merit weight. On each iteration it is determined the amount of new lines that are covered. The variables that were modified either have their weight increased if more lines have been covered or decreased if no new lines have been reached.

All of them share a common framework to generate input variable values that provide a random value within the domain of each variable. This framework ensures that the values provided are limited within the scope of the program. This is especially relevant for variables types that have a very large possible values range, such as integers or floating points to ensure that the input variables values set is of relevance to the test scenario. Currently is up to the user to define these limitations but to improve results and applicability, they should be extracted automatically from an initial code analysis. This could be achieved by introducing a parser for the intermediate representation of the model, available within the toolset.

For each of the algorithms some conditions/configurations were created. For all algorithms, it is configurable the number of input variables to change on each iteration, and the number of cyclic executions to perform. This last configuration is relevant in this type of devices since most of this behavior contains “memory” causing different paths to be followed, using the same inputs, depending on the number of cycles or time passed since the last cycle. Also, before assuming that a set of values should be used as test inputs it is verified if they have been used previously, ensuring that test sets are not repeated. For all also, it is assumed that the initial test set state is the input variables default values, as defined by the user.

The tabu search algorithm also includes the number of iterations that input variables should be tabued after they have been used in a test set and the number of iterations to find an unused test set, ensuring that if the algorithm stalls at a given plateau, it terminates. The evolutionary algorithm uses a k variable, defining the number of new paths followed, to give rewards to variable that produce changes to the code coverage results.

PRELIMINARY RESULTS AND ANALYSIS

The results present here assume the following algorithm configurations. For all the testing algorithms, only one input variable value is modified between iterations. This allows for a better understanding of a specific value change on the program execution, but may delay the identification of the optimal solution. The number of cyclic executions in each iteration was set to two, just to allow some stability in the execution, especially for the interlocking algorithm. The numeric values were limited to the range between zero and ten. The maximum number of cyclic executions was set to five hundred, limiting all algorithms to reach a solution before this maximum. All algorithms were ran one hundred times, their averages calculated, and the conclusions were drawn based on these averages. For the nested conditions and triangle type algorithms, the tabu search algorithm kept the modified variable tabued for just one iteration, due to the small amount of input variables, while for the interlocking algorithm, values were tabued for ten iterations.

The nested conditions algorithm proved to be easily solved by all algorithms. All were able to reach a 100% code coverage before the computational limit. The tabu search algorithm provided the best results, improving the evolutionary algorithm by 33% while the random performed

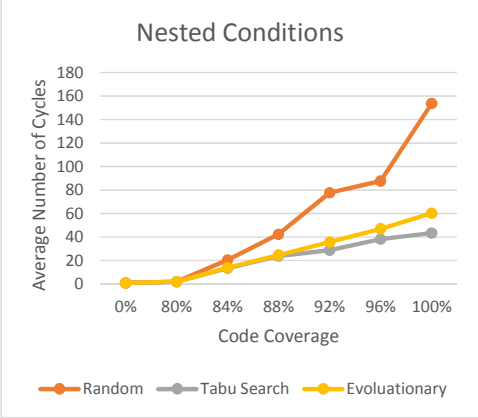


Figure 4: Nested Condition Algorithm Code Coverage Reach

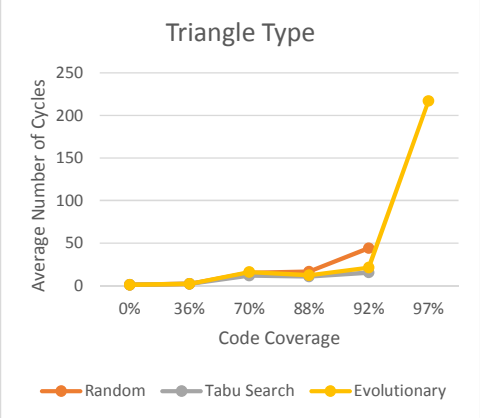


Figure 5: Triangle Type Algorithm Code Coverage Reach

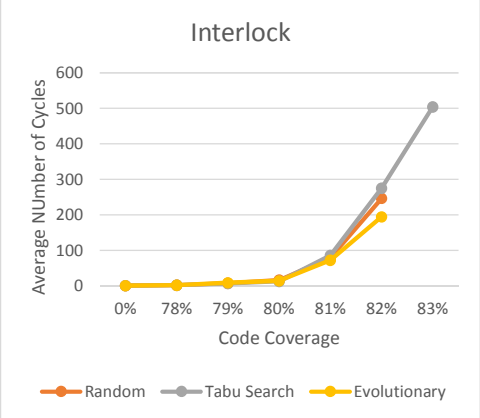


Figure 6: Interlocking Algorithm Code Coverage Reach

112% worse than the tabu search. On the other hand, while the random and the evolutionary algorithm always reached 100% code coverage, the tabu search was only able to reach it 51% of the times due to the algorithm restrictions (tabued element and that no two input sets are alike).
The triangle type algorithm showed a result somewhat different. All algorithms were capable of reaching the 92% code coverage in a similar steady evolution, having the random performed somewhat worse. However, the evolutionary algorithm was the only to be able to reach 97% code coverage although it required a lot more iterations to reach it.
For the interlock algorithm, the results were much more similar between metaheuristics. All of them followed a similar growth pattern in terms of cycles required to reach a higher code coverage even though tabu search was able to reach 83% while all others only reached 82%.
From this observed results is clear that a guided metaheuristic can provide a better solution, within a more limited time, when compared to a random value generating algorithm. But when applied to power systems specific automation behaviors, all methods behave very similarly.

LIMITATIONS TO THE EXPERIMENT

Even though this experiment already provides some useful insights on the applicability of these algorithms for producing test case scenarios, some limitations should be considered in future iterations. First, the code coverage metric is an interesting metric to validate the structural validity of the implementations but is does not provide an actual representation of the normal program execution. Different metrics should be used, like input or output variables range coverage, to improve these algorithms. Second, the testing algorithms do not represent the full range of capabilities of the languages defined in IEC 61131-3. They do not stimulate situations that include real-time timers, control execution flows, recursive invocations or simultaneous program executions, just to name a few examples. Lastly, and since the range of values for numerical values are currently specified by the user, it is possible to have some paths never executed. To minimize this, an intermediate representation parser could be introduced, as stated above, but other means could also be evaluated. Within the power system automation domain there is a very relevant international standard, IEC 61850 [International Electrotechnical Commission 2011], that is currently viewed as the leading standard within the domain. If the behavior is defined within this domain, it is possible to take advantage of the semantics provided the standard and infer on possible values, evolution of values or relations between input variables
To improve the usability of the experimental setup, an additional revision to the simulation platform should also be done. The revision of the current scheduling engine could minimize result calculation times by ensuring temporal minimization.

CONCLUSIONS AND FUTURE WORK

This paper provided a review of current approaches to automatic simulation calibration, highlighting two approaches. The exogenous values calculation methodology rely on data mining or statistical approaches to provide inputs for simulation models based on real-world data. The model analysis methodology works from the inside out, analyzing the model and determining what input variables have an impact on its execution. Based on this review, an experiment was defined to evaluate the application of model analysis to automation systems defined in IEC 61131-3 programming languages, in particular search-based metaheuristics.
The experiment provided a comparison of three metaheuristics, random search, tabu search and evolutionary, when applied to benchmark algorithms and real-world power systems interlocking conditions. It is clear that guided algorithms tend to provide a better, and in less time, solution when applied to the benchmark algorithms. However, when applied to specific power automation algorithms, they all behave in a very similar way disallowing a clear differentiation between metaheuristics.
The application of search based metaheuristics to identify test case scenarios can be of value to this domain, facilitating the testing of PLCs. However, care should be taken to ensure that the testing procedure actually evaluates the real-world execution of the implemented behaviors. The integration of exogenous variables calculations methodologies or other methods or sources of information that can provide semantics to the input variables values, like IEC 61850, could provide a more robust and reliable solution.

REFERENCES

BRAIONE, P., DENARO, G., MATTAVELLI, A., VIVANTI, M. AND MUHAMMAD, A., 2013. Software testing with code-based test generators: data and lessons learned from a case study with an industrial software component. *Software Quality Journal*, 22(2), pp.311–333.

BRAND, K.-P., WIMMER, W. AND LOHMANN, V., 2003. *Substation Automation Handbook*, Bremgarten - Switzerland: Utility Automation Consulting Lohmann.

CARLSSON, H., SVENSSON, B., DANIELSSON, F. AND LENNARTSON, B., 2012. Methods for Reliable Simulation-Based PLC Code Verification. *IEEE Transactions on Industrial Informatics*, 8(2), pp.267–278.

FERREIRA, E., PAULO, R., CRUZ, D. DA AND HENRIQUES, P., 2008. Integration of the ST language in a model-based engineering environment for control systems: An approach for compiler implementation. *Computer Science and Information Systems*, 5, pp.87–101.

FINLEY, J.R., PINTÉR, J.D. AND SATISH, M.G., 1998. Automatic model calibration applying global optimization techniques. *Environmental Modeling & Assessment*, 3(1-2), pp.117–126.

GECCHELE, G., ROSSI, R., GASTALDI, M. AND CAPRINI, A., 2011. Data Mining Methods for Traffic Monitoring Data Analysis: A case study. *Procedia - Social and Behavioral Sciences*, 20, pp.455–464.

HIGDON, D., KENNEDY, M., CAVENDISH, J.C., CAFEO, J.A. AND RYNE, R.D., 2004. Combining Field Data and Computer Simulations for Calibration and Prediction. *SIAM Journal on Scientific Computing*, 26(2), pp.448–466.

HUANG, Z., DU, P., KOSTEREV, D. AND YANG, B., 2009. Application of extended Kalman filter techniques for dynamic model parameter calibration. In *2009 IEEE Power & Energy Society General Meeting*. IEEE, pp. 1–8.

INTERNATIONAL ELECTROTECHNICAL COMMISSION, 2003. *IEC 61131 - Programmable Controllers - Part 3: Programming Languages* 2nd ed., International Electrotechnical Commission.

INTERNATIONAL ELECTROTECHNICAL COMMISSION, 2011. *IEC 61850 - Communication networks and systems for power utility automation* 2nd ed., International Electrotechnical Commission.

JEE, E., YOO, J., CHA, S. AND BAE, D., 2009. A data flow-based structural testing technique for FBD programs. *Information and Software Technology*, 51(7), pp.1131–1139.

JIANG, Z., LI, S. AND LIU, X., 2012. Parameters Calibration of Traffic Simulation Model Based on Data Mining. *Journal of Transportation Systems Engineering and Information Technology*, 12(6), pp.28–33.

JOHN, K.-H. AND TIEGELKAMP, M., 2010. *IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools* 2nd ed., Springer-Verlag Berlin Heidlberg.

KALSI, K., 2012. Model calibration of exciter and PSS using Extended Kalman Filter. In *2012 IEEE Power and Energy Society General Meeting*. IEEE, pp. 1–6.

KIANFAR, J. AND EDARA, P., 2013. A Data Mining Approach to Creating Fundamental Traffic Flow Diagram. *Procedia - Social and Behavioral Sciences*, 104, pp.430–439.

KING, J.C., 1976. Symbolic execution and program testing. *Communications of the ACM*, 19(7), pp.385–394.

LEMOS, M., SANTOS, J.M., VARELA, G., BERNARDO, R., PAULO, R. AND CARRAPATOSO, A., 2011. Substation Automation Systems Current Challenges and Future Requirements - The InPACT Project Perspective. In *21st International Conference on Electricity Distribution, CIRED 2011*. Frankfurt, Germany.

LI, S., SONG, Z., ZHOU, M. AND LU, Y., 2013. Sensor data quality assessment for building simulation model calibration based on automatic differentiation. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, pp. 752–757.

LIU, H., YU, Q., DING, W., NI, D., WANG, H. AND SHANNON, S., 2011. Feasibility study for automatic calibration of transportation simulation models. , pp.87–94.

MCMINN, P., 2004. Search-based Software Test Data Generation: A Survey. *Software Testing, Verification & Reliability*, 14(2), pp.105–156.

MILLER, W. AND SPOONER, D.L., 1976. Automatic Generation of Floating-Point Test Data. *IEEE Transactions on Software Engineering*, SE-2(3), pp.223–226.

NEW, J., SANYAL, J., BHANDARI, M. AND SHRESTHA, S., 2012. *Autotune E+ Building Energy Models*,

PAPAEFTHYMIOU, G. AND KLOCKL, B., 2008. MCMC for Wind Power Simulation. *IEEE Transactions on Energy Conversion*, 23(1), pp.234–240.

PARK, S., PARK, C. AND WANG, G., 2008. PLCStudio: simulation based PLC code verification. *2008 Winter Simulation Conference*, pp.222–228.

PEREIRA, A., LIMA, C. AND MARTINS, J.F., 2011. The use of IEC 61131-3 to enhance PLC control and Matlab/Simulink process simulations. In *2011 IEEE International Symposium on Industrial Electronics*. IEEE, pp. 1243–1247.

WANG, F., HASBANI, J.-G., WANG, X. AND MARCEAU, D.J., 2011. Identifying dominant factors for the calibration of a land-use cellular automata model using Rough Set Theory. *Computers, Environment and Urban Systems*, 35(2), pp.116–125.

WANG, R. ET AL., 2012. Modeling and Validation of PLC-Controlled Systems: A Case Study. In *2012 Sixth International Symposium on Theoretical Aspects of Software Engineering*. IEEE, pp. 161–166.

WILSON, T., 1999. PLC Based Substation Automation And SCADA Systems And Selecting a Control System Integrator. In *Western Electric Power Institute Distribution/Substation Automation Workshop*. Las Vegas, Nevada, USA.

WU, Y.-C. AND FAN, C.-F., 2014. Automatic Test Case Generation for Structural Testing of Function Block Diagrams. *Information and Software Technology*.

ZAN YANG, MIN, B. AND GWAN CHOI, 2001. Simulation using code-perturbation: black- and white-box approach. In *Proceedings of the IEEE 2001. 2nd International Symposium on Quality Electronic Design*. IEEE Comput. Soc, pp. 44–49.