TESTING & VERIFICATION OF PLC CODE FOR PROCESS CONTROL

B. Fernández, E. Blanco, A. Merezhin, CERN, Geneva, Switzerland

Abstract

Functional testing of PLC programs has been historically a challenging task for control systems engineers. This paper presents the analysis of different mechanisms for testing PLC programs developed within the UNICOS (UNified Industrial COntrol System) framework. The framework holds a library of objects, which are represented as Function Blocks in the PLC application. When a new object is added to the library or a correction of an existing one is required, exhaustive validation of the PLC code is needed. Testing and formal verification are two distinct approaches selected for eliminating failures of UNICOS objects. Testing is usually done manually or automatically by developing scripts at the supervision layer using the real control infrastructure. Formal verification proves the correctness of the system by checking whether a formal model of the system satisfies the properties or requirements. The advantages and limitations of both approaches are presented and illustrated with a case study, validating a specific UNICOS object.

INTRODUCTION

Nowadays any software functionality is required to be delivered faster and with minimum cost while maintaining the quality expected. This applies to any software and also to process automation applications. These critical applications needs to be extensively tested to validate the requirements and ensure a smooth execution 24/7 in industrial plants as manufacturing, nuclear plants, pharma, cooling systems, etc.

Generally, it is accepted to divide tests according to their level of specificity into: (1) unit testing, where a specific section of code is tested separately, (2) integration testing, where all individual units are put together to be checked globally and (3) system testing where the complete system is tested as a whole application.

Currently, testing is applied to process control applications as a black box tests of function blocks or modules in combination with input simulations and visualization of the results by the manual tester. Test cases are manually created from the requirements or specifications by the developer and basically testing is considered as another activity of development. Following the industrial automation standard ISA-62381 [1], process control applications testing is done in two main stages: Factory and Site Acceptance Tests, FAT and SAT respectively. The difference being whether the tests are performed in the factory or in the real plant.

Automated tests allow a better implementation of repetitive or tedious duties and can be hardly accomplished by a manual tester [2]. These tests are a complement and not a replacement of the tests procedures done by hand, indeed not all test are easily automated. The manual tester's analytic skills can hardly be replaced and these testers are frequently needed in the interpretation of the test results. Therefore manual and automated tests are not unrelated but complement each other.

Formal methods, within this context, are mathematical techniques for the specification, development and verification of software and hardware systems. Applied to programming code, formal methods provide a precise semantics of a program. Having this formalization we can prove that the program is correct, meaning it meets the user specifications. Considering that other testing mechanisms can never guarantee error-free applications this technique appears to be a solution, however the complexity is an evident drawback.

The paper describes the environment where testing procedures must be applied, the test methods employed focusing basically in automated tests mechanisms and formal methods. Finally a summary of results and analysis of the advantages and disadvantages of applying these mechanisms is depicted.

UNICOS PLC CONTROL SYSTEMS

UNICOS is an object-based framework providing a methodology and a set of tools to develop industrial control systems. This framework is used in industrial installations, e.g. the LHC (*Large Hadron Collider*) cryogenic control systems, cooling and ventilation control systems, and the LHC vacuum control systems. This framework, which has been developed at CERN over the past 10 years, contains several packages, according to the kind of control system to be developed. The UNICOS-CPC package [3] is devoted to process control systems. The control systems studied in this paper are composed by two layers: the control layer based on PLC (*Programmable Logic Controller*), and the supervision layer based on a SCADA (*Supervisory Control And Data Acquisition*).

UNICOS Object Library

In the framework, each physical device (e.g. valve) and process units are modeled as UNICOS objects, the complete set of objects is called the *Baseline* library.

Each new object to be designed under the UNICOS framework must follow the UNICOS metamodel [4]. This metamodel defines a predefined structure with a naming convention, thus the interface of the new object is defined. However the object functionality is not expressed and validated at this stage.

ISBN 978-3-95450-139-7

This object library is composed by Function Blocks in the PLC developed with the *ST* (Structured Text) programming language from the IEC 61131-3 standard [5].

The specification of the objects functionality is currently described in a non-formalized way. It consists in well-structured documents where the developer describes the object functionality. In addition, test catalogs describing the behavior of some parts of the object have been created for testing purposes.

Many authors have arrived to the conclusion that most of the failures in a system are due to incomplete and unambiguous requirements specification.

In UNICOS, an initial analysis of the functional requirements that we want to test and verify has been made. This analysis is the very first step to formalize and complete the UNICOS objects specification.

TESTING & FORMAL VERIFICATION APPROACHES

Manual Testing & Automatic Testing

The straightforward approach to test PLC programs is to do it manually using the supplier's IDE connected to PLC. Within the UNICOS framework a PLC object is normally meant to have a supervision counterpart representation. Having this in mind, another possible approach would be using a SCADA application connected to the PLC. Assuming the communication is correct, a tester can manipulate devices and observe their reaction. Here, the good practice would be to have a test catalog with all the test cases explicitly written. The tester can simply follow the instruction and mark performed tests with the corresponding results.

The UNICOS Baseline has been developed containing different object parametrization. To complete the tests additional PLC code has been programmed to test some object features which require additional inputs and/or variables (e.g. interlocks behavior). Table 1 shows a simplified example of test catalog for three UNICOS object configurations which validates the behavior on temporal stop interlock (TS). The table's header represents the test sequence performed at SCADA layer. Each line represents expected states of the corresponding device parametrization. The test script has been based on the test catalog.

Table 1: Test Catalog Example: A Sequence

	Initial state	TS goes on	TS goes off
Config 1	off	off	off
Config 2	on	off	on
Config 3	100	0	100

The next evolutionary step was to automatize the tester's routine. The SCADA programming language really fitted this need, as its main purpose is to send an order and retrieve a result. The test script appeared to be simple

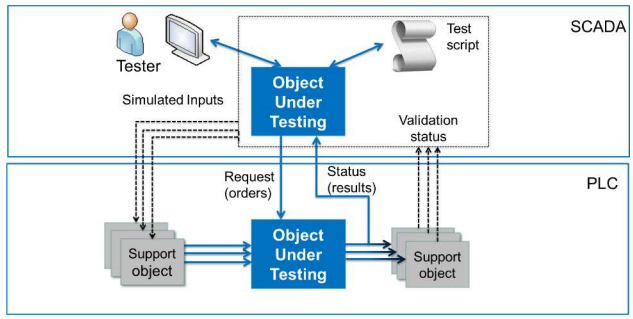

Figure 1: Testing topology.

and laconic though the execution time was rather long and the maintenance has been found complicated. This kind of automated testing can be classified as a system testing according to the topology (see Fig. 1) and since the test script is at the supervision level. More appropriate unit testing could be done by using proprietary simulator tools thought this solution can be used independently of the PLC platform (Siemens, Schneider or Codesys). The developed test scripts contain a certain level of abstraction (implemented as test scenarios) allowing them to be reused for different UNICOS object instances.

Formal Verification

A different approach is to apply formal verification to the UNICOS-CPC baseline objects. Formal verification depends on the use of mathematically based techniques and tools for the design, development and analysis of systems. The reason of using formal methods is to try to increase the level of correctness. Using testing approaches is usually impossible to test all the possible combinations, this problem can be solved by using formal verification and bugs can be detected before the system is implemented.

The general idea is to create a formal model of the real system and check its correctness against formal requirements describing the expected behavior of the system. There are two different approaches of verification: axiomatic verification and algorithmic verification.

Axiomatic verification is a formal method for verifying the functional correctness of a structured sequential program by tracing its state changes from an initial condition to a final condition according to a set of rules. Theorem Provers are the tools used for this kind of verification (e.g. Coq theorem prover). It is really hard to automatize and the success of this technique depends also on the skills of the engineers and researchers. It has been applied to operating systems and compilers but also to PLC programs. In [6], the *Coq* theorem prover is applied to formalize the semantics of the PLC languages described in the standard IEC 61131-3.

Algorithmic verification is usually referred as *model checking*. This technique use semi-algorithms to check that a global model (representing the whole system) meets the requirements. It is feasible to automatize the creation of

ISBN 978-3-95450-139-7

these model and some theoretical limitations are overcome by using abstractions. It is applied mainly to hardware, abstract software and systems models.

Most of the verification applied in industry is done using algorithmic verification and this is the approach selected to verify PLC program based on UNICOS. Applying model checking to any system requires the formalization of the system and also of the requirements (See Fig. 2):

1. Formalization of the system is done by creating models using formalisms as Petri nets, labelled transtion systems, timed automata, hybrid automata, etc.

2. The requirements are usually formalized using temporal logic. Temporal logic is a formalism to express unambiguously the requirements or properties. It contains a set of rules for representing propositions qualified in terms of time.

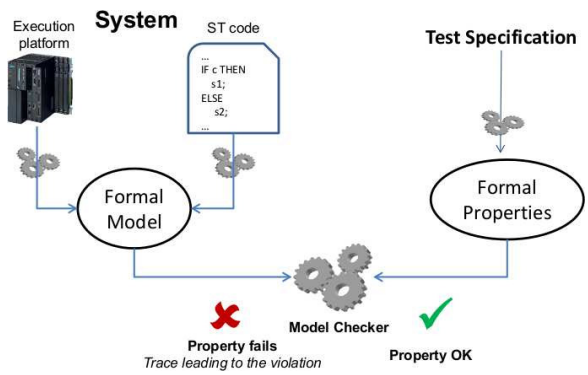

Figure 2: Model checking approach.

There are a significative number of available model checking tools available. Some of the most popular or relevant are NuSMV, UPPAAL or SPIN.

Model checking approaches for the verification of PLC programs have been studied before for different application areas and description languages. The reference [7] applies algorithmic verification to the SFC language. Untimed SFC models are transformed into the input language of the Cadence SMV tool. Timed SFC models are transformed into timed automata and they can be analyzed by the UPPAAL tool. The reference [8] models and analyzes a safety-critical system using coloured Petri nets as formalism and symbolic model checking for the analysis. In [9] they model and apply formal verification to a particular timed multitask PLC to a specific platform from the Bosch Group.

Our Approach and Contribution For the verification of UNICOS-CPC baseline objects, we propose a methodology for its formalization in automata based models for further treatment with verification tools (see Fig. 3). Baseline objects are formalized in a generic intermediate model and then using another transformation

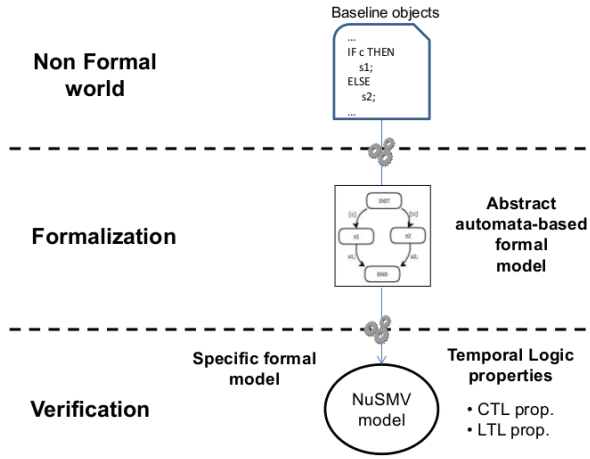

Figure 3: Transformation of ST PLC code to NuSMV input language through an automata based abstract model.

formal verification can be applied with NuSMV (as shown in the figure) and other model checkers, like UPPAAL.

This methodology in being currently implemented in an automatic generation tool based on *EMF* (Eclipse Modeling Framework) and *Xtext*, from ST we can produce formal models for NuSMV, UPPAAL and the input language of the BIP (Behaviour, Interaction, Priority) framework, passing through the intermediate model. This approach will provide a good test bench to compare performance limitations and identify the more appropriate tool for the kind of properties to verify.

Formal methods techniques can be applied also to complete control systems developed with UNICOS. In the paper [10] we presented a first approach for formalizing a complete UNICOS control system using the BIP framework.

ANALYSIS & EVALUATION OF THE APPROACHES

PLC software development lacks of modern software engineering best practices such as unit test or daily builds.

Automated testing via SCADA is feasible and in certain cases allows to save the time spend on testing. We found examples where the test case can be shared between different objects. We also reuse automated test scripts to test different types of PLC. Automatic testing permits reducing human errors, though the approach has several issues: *(1)* it involves many components (the systems must be operative and properly communicating) which makes it hard to support and replicate; *(2)* it is not possible to test all possible combinations of parameters of a baseline object; the cost of each parameter's combination test is relatively high; *(3)* it is a black box testing of PLC code - it is not possible to manipulate or observe the variables which are not communicated. Additional PLC code must also be implemented to simulate PLC events and inputs via the supervision scripts.

Formal verification techniques are not widely used in

ISBN 978-3-95450-139-7

Table 2: Overview of Automatic Testing and Formal Verification for UNICOS Baseline Objects

	Advantages	Disadvantages
Automatic Testing	Testing with the real system Technology is available Reduce human errors Reusable for different PLC	Sophisticated maintenance High price for new test case Black box testing Difficult to find the source of the problem
Model Checking	Explores all the combinations Earlier bug detection Avoid human errors Complexity hidden by the generation tools Counter-examples to find the source problem	Verification of a system model Need of automatic generation tools Need to prove the transformations State space explosion Applying abstractions techniques is not trivial

industry because of the complexity of, both, building the models of the PLC systems and formalizing the specifications. Control system engineers are usually not familiar with these techniques, rising the need of automatic generation tools to hide this complexity. Another common problem in formal verification is the state explosion problem. As an example, trying to verify UNICOS-CPC objects with about 750 lines of ST code and about 120 variables, integers or float, causes some severe issues with the performance of the model checker. However, providing *(1)* automatic generation tools for the models, *(2)* a good language to express properties, and *(3)* the right technique for abstraction, model checking can be an useful and powerful technique to verify PLC control programs. Model checkers provide counter-examples, when the model does not reach a specification property. This feature definitely helps the developer to find the original problem.

Table 2 summarizes some of the advantages and disadvantages of both approaches.

CONCLUSIONS AND FUTURE WORK

This paper present an analysis of two different techniques aimed to test and verification PLC control systems, specifically applied to UNICOS-CPC baseline objects. Automatic testing and formal verification approaches are meant to guarantee that the PLC control systems meet the specification requirements. Both techniques have different advantages and disadvantages but they can complement each other. While automatic testing seems a more feasible technique to implement, formal verification seems a powerful future option.

Ideally formal verification could be integrated in the development process of PLC control systems, in our case inside the UNICOS framework. This will help the control engineers to detect and correct bugs before the PLC application is deployed. Once this verification step is finished, automatic testing could be applied within the real PLC control system, then bugs can be detected if any situation was not taken into account on the formal models.

Next work on automatic testing will focus on extending the test coverage and also performance improvement when making the tests.

Concerning formal verification, an automatic generation tool to produce models of our control systems and verify them with different model checkers is under development. A mathematical proof of the transformations rules (semantics preservation) is required as providing a formalism to specify unambiguously the test catalog, hiding the complexity of temporal logic for control engineers.

REFERENCES

[1] ANSI/ISA 62381: Factory Acceptance Test (FAT), Site Acceptance Test (SAT), and Site Integration Test (SIT), Automation Systems in the Process Industry.

[2] E. Dustin, T. Garrett, B. Gauf. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality". *Addison-Wesley Professional*, March 2009.

[3] E. Blanco et al. UNICOS Evolution: CPC version 6, *ICALEPCS*, Grenoble (France), 2011.

[4] B. Copy et al. Model Oriented Application Generation for Industrial Control Systems, *ICALEPCS*, Grenoble (France), 2011.

[5] IEC 61131: Programming languages for programmable logic controllers, International Electrotechnical Commission.

[6] J. O. Blech and S. O. Biha. Verification of PLC Properties Based on Formal Semantics in Coq, 9th SEFM, Montevideo (Uruguay), 2011.

[7] N. Bauer, S. Engell, R. Huuck, B. Lukoschus, and O. Stursberg. Verification of PLC programs given as sequential function charts. Integration of Software Specification Techniques for Applications in Eng., Springer, LNCS, 2004.

[8] T. Bartha et al. Verification of an Industrial Safety Function using Coloured Petri Nets and Model Checking, 14th MITIP 2012.

[9] H. B. Mokadem et al. Verification of a Timed Multitask System with UPPAAL, IEEE Transactions on Automations Science and Engineering 7, 4(2010) 921-932.

[10] B. Fernández et al. Model-based Automated Testing of Critical PLC Programs.*11th INDIN*. Bochum (Germany), 29-31 July 2013.

ISBN 978-3-95450-139-7