UDC 004.45 **doi: 10.26906/SUNZ.2023.1.051**

Vasyl Vasiuta, Artem Danyleiko, Viktoriia Vasiuta

National University «Yuri Kondratyuk Poltava Polytechnic», Poltava, Ukraine

CONCEPTUAL APPROACHES OF ORGANIZING PROCESSES OF VERIFICATION OF SOFTWARE PROJECTS OF CRITICAL INFORMATION-CONTROLLED SYSTEMS ON THE BASIS OF REQUIREMENTS FOR THEIR LIFE CYCLE

Abstract. The implementation of an effective technology to ensure the required level of quality of software systems is one of the urgent and important tasks of software engineering. It is especially important to solve this problem for critical information management Ukrainian nuclear power plants. Ensuring the required quality of software systems requires an understanding of the process of organizing the verification of software projects. The purpose of this study is to review the main conceptual approaches to the organization of verification processes for software projects of critical information management systems based on the requirements for their life cycle. To achieve this goal, the authors considered the automation of Ukrainian nuclear power plants in the form of software and hardware complexes and the general principles for the implementation of critical information management systems based on programmable logic controllers, analyzed the processes of verifying the user's applied logic in these systems, reviewed the main procedures for verifying the applied logic of a special RadiCS platforms.

Keywords: software and hardware complex, critical information management system, programmable logic controller, verification.

Introduction

At present, the creation of high-quality software is one of the most important tasks in the development of science and production. The viability of the system ultimately depends on how successfully the software is made. However, due to the fact that only some of the essential properties of software can be directly measured and quantified, ensuring an appropriate level of quality is of paramount importance.

Thus, one of the urgent and important tasks of software engineering is the implementation of an effective technology to ensure the required level of quality of software systems. The solution of this problem is especially important for critical information management systems of domestic nuclear power plants. To ensure the high quality of software systems, it is necessary to have an understanding of the process of organizing the verification of software projects.

The purpose of the study is to review the main conceptual approaches to the organization of verification processes for software projects of critical information management systems based on the requirements for their life cycle.

To achieve the set goals it is necessary:

- consider the automation of nuclear power plants with the help of software and hardware systems;
- to study the general principles for the implementation of critical information-controlled systems based on programmable logic controllers;
- to analyze the processes of verification of the user's applied logic in critical information management systems based on programmable logic controllers;
- review the main procedures for verifying the application logic of the RadiCS special platform.

The main part of the article

Automation of nuclear power plants (NPP) by means of software and technical complexes. Software and hardware complexes (SHC) belong to automated systems. An automated system is a system consisting of personnel and a complex of means of automating its activity, which implements information technology for the performance of established functions. The software and hardware complex of automatic regulation, unloading and limited power of the reactor and accelerated warning protection (SHC ARP-ULP-AWP) using the RadICS platform, intended for use as a technical base for the reconstruction of existing and creation of new control and protection systems with the VVER reactor.

The hardware platform of the ARP-ULP-AWP system and PTK was developed in accordance with the current rules and recommendations of the International Electrotechnical Commission (IEC), the International Atomic Energy Agency (IAEA) and the National Research Council of the USA. Continuous diagnostics of the system is provided by integrated diagnostic devices, which are serviced regardless of the working scheme of security and unified coding.

SHC performs the following main functions:

– automatic regulation of the neutron power of the reactor and/or pressure in the main steam collector of the turbine of the NPP power unit;

– deloading and limitation of reactor power at the levels correspond to the volume of the main technological equipment of the NPP power unit included in the work;

– accelerated warning protection (quick discharge of the reactor to 40-50% capacity in 3-4 seconds) in case of unauthorized shutdown of the equipment.

To increase the reliability of protection operation, three levels of output signal generation based on the "two out of three" majority logic are implemented in the SHC ARP-ULP-AWP. If a licensing procedure is required, the system can be designed in a two-out-of-four configuration according to quality and reliability parameters. When designing the SHC ARP-ULP-AWP, the principles of dividing the functions of the complex into control and protection were initially laid down.

Different groups of functions are implemented in separate, galvanically separated sub-blocks, which increases the operational reliability of the complex as a whole.

General principles for the implementation of critical information management system (IMS) based on programmable logic controllers (PLC). One of the options for using PLC is the field of industrial automation of various technological processes in large and small enterprises. The popularity of controllers is clear. Their use greatly simplifies the creation and operation of both complex automated systems and individual devices, including household ones. The PLC allows to reduce the development stage, simplifies the process of installation and debugging due to the standardization of individual hardware and software components, and also provides increased reliability during operation, convenient repair and upgrade if necessary [1].

It is generally accepted that the task of creating a prototype of a modern PLC appeared in the late 60s of the last century. In particular, in 1968 it was formulated by the leading specialists of General Motors. Then the company was trying to find a replacement for a complex relay control system. According to the received design assignment, the new control system must meet the following criteria:

– simple and convenient creation of technological programs;

- the ability to change the working control program without interfering with the system itself;

– simple and inexpensive maintenance;

– increased reliability at a reduced cost compared to similar relay systems.

Subsequent developments at General Motors, Allen-Bradley and other companies led to the creation of a microcontroller-based control system that analyzed input signals from process sensors and controlled actuator actuators.

Subsequently, the term PLC (Programmable Logic Controller, PLC) was defined in the EN 61131 (IEC 61131) standards. PLC is a unified digital control electronic system specially designed for use in industrial environments. The PLC constantly monitors the state of the input devices and makes decisions based on the user program to control the state of the source devices. The composition and principle of action are shown in Fig. 1.

A simplified representation of the composition and principle of operation of the PLC is well shown in Fig. 1. It can be seen from it that the PLC has three main sections:

- incoming;
- outlet;
- central.

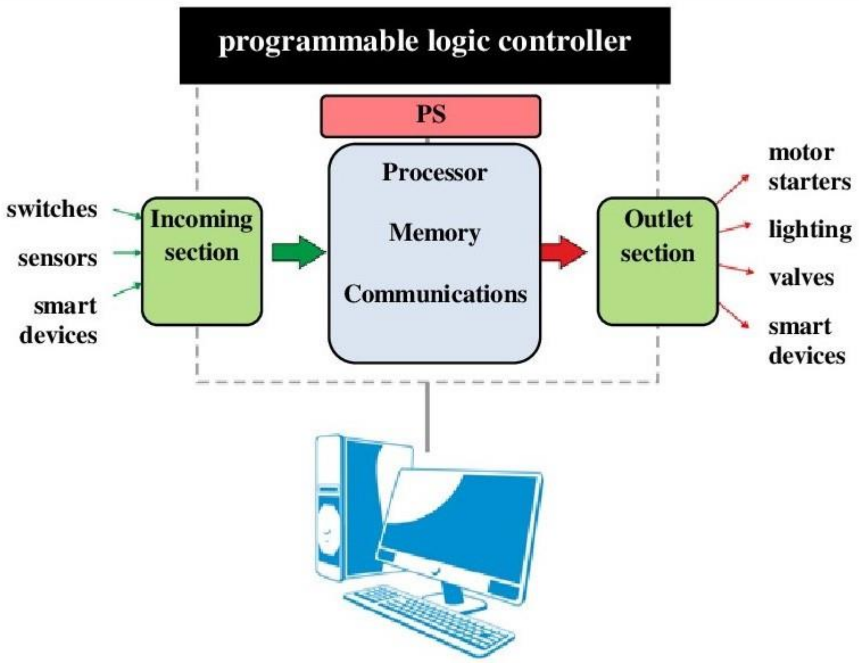


Fig. 1. Composition and principle of operation of PLC

The central section contains a central processor (CP), memory and a communications system. It processes the data received from the incoming data sections and transmits the processing results to the outlet section. It should be noted right away that in large PLC, in addition to the CP operating in the "master" mode, there may be additional "slave" PLC with their own CP. Standard microprocessors (MP) are used as the CPU of a small PLC. Usually, 8- and 16-bit MP fully cope with all standard tasks. But, as noted in IEC 61131, the choice of a specific MP still depends on the tasks assigned to this type of PLC.

To transfer data to another PLC or to connect to PROFIBUS, Industrial Ethernet, AS-Interface data transmission networks in distributed control systems today, communication processors such as DP83867IR manufactured by Texas Instruments (TI) are used.

The input section of the PLC provides input to the central section of the status of switches, sensors and smart devices. Through the output section, the CP controls external executive devices, which may include electromagnetic motor starters, light sources, valves, and smart devices.

Modern PLC using innovative technologies have come a long way from the first simplified implementations of industrial controllers, but the universal principles embedded in the control system have been standardized and are being successfully developed on the basis of the latest technologies.

Today, the largest PLC manufacturers in the world are Siemens AG, Allen-Bradley, Rockwell Automation, Schneider Electric, and Omron. In addition to them, PLC are also produced by many other manufacturers.

According to the design, PLC are divided into monoblock and modular. A fixed set of inputs/outputs is placed in the body of a monoblock PLC along with the CP, memory and power supply. In modular PLC, separately installed input/output modules are used. According to the requirements of IEC 61131, their type and quantity may change depending on the task and be updated over time.

Monoblock functionally complete PLCs can include a small display and control buttons. The display is designed to display current operating parameters and is entered using the buttons of operating program commands and technological settings. More complex PLC are combined from separate functional modules, together they are fixed on a standard mounting rail. Depending on the number of serviced inputs and outputs, the required number of input and output modules is set.

The power supply can be built into the main PLC unit, but more often it is made in the form of a separate power supply unit (PS), which is attached to a row on a standard rail.

The initial source for the PS is most often the industrial network 24/48/110/220/400 V, 50 Hz. Other PSU models can be used as a primary source of constant voltage at 24/48/125 V. PS output voltages of 12, 24, and 48 V are standard for industrial equipment and PLC. In systems of increased reliability, it is possible to install two special reserved PS for duplicating the power supply .

The basic principle of PLC operation is cyclical work, in which the controller executes separate commands one after the other in the sequence in which they are written in the program. At the beginning of each cycle, the program reads the state of the controller's input and writes them down. After executing all the commands and determining the count of the state of the outputs relevant for the given situation, the controller writes the state of the outputs into the memory, which is a table of the state of the process outputs, and the operating system sets the corresponding signals to the outputs that control the actuators. Consequently, all signal combinations are fed into the input module of the controller, and the program monitors their pattern and reacts to changes in the output states based on the built-in algorithm.

Analysis of user application logic verification processes in PLC-based information and control systems. The reliability and security of critical information management systems largely depend on the quality of the software that performs critical functions. Latent bugs (bugs not discovered during testing and validation) in critical software are risk factors for system failure.

An independent review of critical software that confirms the performance of the declared functions and provides an assessment of the likelihood of hidden errors is a necessary condition for regulatory requirements for various industries.

From this point of view, the main problems are: the reliability of independent verification, the assessment of the reliability of latent defects, the completeness of test coverage for critical software and, therefore, the quantitative assessment of functional safety.

Independent review and validation is a key method for qualifying critical software. Implementation is a legal requirement in critical areas such as nuclear power ("Software for Computerized Systems Important to Safety in Nuclear Power Plants" - IAEA Safety Standards Series) and the space industry (ECSS-Q-40B, ECSS-Q-80B)) other.

An independent review as part of the ICS software qualification tests determines the actual ability to guarantee the required level of security and quality of ICS for critical programs in general.

The work of ISO is based on procedures known as ISO/IEC. Translation from English "Verification" gives it a clear interpretation: Verification - Verification [2]. To help you understand this, here is an example of a typical test: testing a program or a test device. Tests are carried out in accordance with certain requirements for the item being tested and it is recorded that the requirements are met. The result of the check is the answer to the question "Does the object meet the requirements?".

Checking allows for timely corrections and warnings.

Measures to eliminate inconsistencies have been identified to avoid or minimize claims from external and internal consumers, to improve the operating conditions and use of the test element.

Hidden bugs in critical software are a major risk factor for emergencies throughout the system. Independent verification of critical software is a mandatory requirement of the international regulatory framework.

The purpose of verification is to ensure that an object (request or program code) is verified and implemented without unexpected features.

There are two main methods for validating and analyzing systems in validation and testing processes: validation and automated analysis are static methods that can be performed at all stages of the system development process, and testing is a dynamic method that is performed when the program is already created in place, then is at the stage of implementation of the system and after its completion.

Testing is the process of running a program (or part of a program) in order to find errors. Debugging is not a type of testing. Although the words debugging and testing are often used interchangeably, they refer to different activities. Testing is an activity aimed at identifying errors.

The purpose of a correction is to determine the exact nature of a known error and then correct the error. These two activities are related - the test results are the input for debugging. All proofs of system safety are based on the following assumption: The number of faults in the system leading to emergency situations is much less than the total number of faults in the system. Security should be focused on identifying potentially dangerous bugs. If it is found that these errors do not appear or do not appear but have no serious consequences, the system is considered reliable. The necessary prerequisites for a high level of reliability and safety of critical software are the availability of appropriate (effective) regulatory and methodological support, as well as the widespread use of tools to support qualification testing (expertise) processes, which reflects the modern dynamics of standardization in the development of information technology and software. The main direction of improving the reliability of ICS software quality assessment for critical use in qualification tests is the diversification of verification technologies.

The fulfillment of these conditions determines the real possibilities to guarantee the necessary level of security and quality of the entire information and control system, as well as within the framework of risk-based security regulation approaches.

The quality of ICS software for critical programs, given the risks of hidden software errors, is an urgent part of the implementation of approaches to safety regulation and qualification of ICS for critical programs in various fields of application (nuclear power plants, space, transport, etc.).

The first stage of the review is to perform a static code analysis (SCA) and code review (code review - CR) in the SW language. This code can describe an electronic design (ED) or its functionally complete components intended for the configuration of integrated circuits FPGA, CPLD, ASIC, for example, Function Block Library (hereinafter - ED). The SCA and CR method for a SW code (hereinafter referred to as SW-SCA/CR) is used to check whether the analyzed code complies with the encoding rules.

The SW SCA/CR process is used to verify that there are no coding errors and to ensure the efficiency of electronic design development.

The SCA/CR-SW procedure is applicable to an ED for which the coding step [3] has been completed.

Composition of the independent verification team for the SCA/CR-SW.

The SW SCA / CR process is carried out by an independent group of verification specialists (hereinafter referred to as the group) who did not participate in the development of IDEs, who have the necessary knowledge, skills and experience in developing (synthesis) and reviewing the code of functionally complex IDEs.

The ED code is generated in the design environment (tool) for a specific type of FPGA. The certificate(s) must be available for such tool (IC) and, if possible, the IC approval made when selecting an IC for an IDE design is not part of the described SW SCA/CR processes in this procedure. Functional safety data package certified by an independent certification body - TÜV Rheinland Group.

After creation (synthesis, development), preliminary simulation of the functional test and debugging of the specified ED code, the following is submitted to the verification team leader for direct SW SCA / CR in the specified order as decided by the IDE development team leader:

- a complete set of valid technical design documents that will be used as input for the development of the IDE code (including specifications for product development, product concept, product architecture description);
- a list and a brief description of the functional libraries of plug-ins by developers of the ED code in the ED design environment and a description of the procedure for connecting (disconnecting) these libraries when developing code subject to SW SCA;
- A complete electronic copy of the functional plugin libraries used by ED code developers in the ED design environment when developing code subject to SW SCA.

Preliminary ED design submission (e.g. Architecture Description (AD) or Detailed Description (DD) submission) is an optional step in the SCA/CR-SW procedure when the developer submits the ED design informs the SCA/verifiers. CR SW. The verification team leader is responsible for identifying the needs and planning for this phase.

The pre-submission of the IDE project should be done taking into account the large size and complexity of the IDE project and the lack of knowledge of the reviewers.

An IDE project preview should always be performed (including a resubmission) whenever new documents are created or changes are made to the original documents required by SW SCA/CR.

The initial criteria for the SW SCA/CR level is approval by the auditors for a satisfactory meeting, with a preliminary presentation of the draft IDE in the form of minutes of the meeting drawn up by the audit team leader.

Having received all accompanying and descriptive documents for the developed IDE code (in paper and/or electronic form) and complete, exact electronic copies of the developed IDE code, approved by the appropriate officials, and everything is in development with a design environment associated with the functional libraries environment code (in electronic form) the verification team leader executes the detailed plan in strict accordance with the overall verification and validation plan and allocates roles and responsibilities for the direct implementation of SW SCA / CR in the member team.

Planning consists of identifying the units underlying the SCA/CR SW and determining the time required to complete the SCA/CR SW. The SCA/CR-SW schedule should include the date and time of the

SCA/CR-SW and the composition of the verification team.

The planning phase ends when the following criteria are met:

- SW SCA/CR plan developed;

- If it is not possible to determine the exact dates of some phases of the plan, SW SCA/CR should indicate the limits of possible dates.

- The SCA/CR-SW plan indicates sufficient resources.

- compiled SW schedule;

- Responsibilities are distributed among auditors.

The primary responsibility for SW CR implementation lies with the verifiers.

The input criteria for the SW CR verifier are:

- the pre-submission of the IDE project, if necessary, has been successfully completed;

- descriptive documentation for the developed IDE code;

- necessary carrier materials are available for carrying out SW CR.

- The time required to complete the SW CR has been agreed with the test centers.

Tasks that need to be solved when performing SW CR:

1. Analyze the SW SCA results by examining the violations detected by the HD Designer IC.

2. Each examiner must indicate how long it takes to complete the SW-CR.

The start-up phase of the SW CR is completed when the following criteria are met:

– Each examiner completed an SW CR confirmation letter.

- Prepared questions, a list of inconsistencies and/or deficiencies identified in this IDE Code, and comments on the results of the SW SCA.

Analysis of SW SCA/CR results

During the meeting, which analyzes the results of the SCA/CR-SW, discusses the problems, the auditors identify nonconformities and/or deficiencies and decide on the degree of importance of the identified nonconformities and/or deficiencies. When discussing submissions that are subject to the SCA/CR SW, verifiers may identify new inconsistencies and/or deficiencies not identified during the SW CR. Options for addressing identified nonconformities and/or errors should not be discussed when reviewing SW SCA/CR results.

Responsibility for the formation of the results of the SW SCA / CR implementation rests with all verifiers.

The introduction criteria for a successful SW CR review meeting are:

- verifiers performing SW CR are fully present;

– Descriptive documents for the developed ED code were available for SW CR implementation.

- definition of the goals of the meeting to analyze the results of the implementation of SW SCA / CR.

Tasks that need to be solved during the meeting to analyze the results of code review:

1. A short introduction by the verification team leader, in which he should:

— ensure that all test facilities performing SW CR are aware of their responsibilities;

- if necessary, inform the members of the audit team about the structure of the SW SCA / CR procedure (Fig. 1) and the place of the SW CR in it.

2. Review of the level of preparation of examiners by the head of the examination group, consisting of:

— asking auditors about the time they spent completing the SW CR and the number and main types of errors and/or nonconformities identified in the review;

- Decide that the SW CR review session builds on all previously prepared comments and covers all aspects of the review.

3. Read the list of materials provided by the reader for SW CR.

4. Identification of inconsistencies and / or shortcomings by auditors:

- identify inconsistencies and / or defects based on the results of the SWCR;

- compare the obtained results of SW SCA with the results of SW CR and identify inconsistencies and / or errors for their further elimination (priority (significance) of the consequences of the identified inconsistencies of the specified safety requirements, which should be determined);

- assess the degree of impact of inconsistencies and / or shortcomings identified during the implementation of SW SCA / CR on other documents (organizations) previously created within the framework of the relevant project, and determine the need for their changes;

- If the auditors decide that the inconsistencies and / or shortcomings identified during the implementation of SW SCA / CR lead to a change in previously created documents (organizations) within the specific project under consideration, the head of the audit team initiates the change and processes it in accordance with the method described in the internal instructions;

- Do not discuss the style or options for correcting identified inconsistencies and / or shortcomings.

5. Recording of all disputes and/or deficiencies by the registrar, who must:

- indicate any detected inconsistencies and / or defects, indicating their place in the IDE code in question and the number of the violated rule;

- during or after the completion of the meeting to analyze the results of the implementation of the QA software, check the identified inconsistencies and / or shortcomings, in order to then submit them to the auditors for approval;

- record the time spent in the meeting to analyze the results of the SWCR implementation.

6. Determine the status of the ED code after making one of two possible decisions:

- if no nonconformities and/or deficiencies are identified, the SW CR result is approved as positive, and the verification team leader prepares a general report for SW SCA / CR regarding the procedure carried out and the results achieved;

- If inconsistencies and/or deficiencies are identified, a decision is made to further transfer the IDE code to the IDE development team for correction, and then a re-SCA / CR SW is organized and the results of the intermediate analysis are turned into the results of the SW CR implementation in the form of an interim report or draft final reports.

The SW CR implementation meeting ends when the following criteria are met:

- if the ED code has passed the SW-CR procedure in the intended scope;

- inconsistencies and / or shortcomings were recorded, their location is clearly indicated in this material and this list is approved by the reviewers.

– If nonconformities and/or deficiencies are identified, the verification team leader presents the ED code and a list of nonconformities and/or deficiencies based on the results of the SCA/CR-SW to the ED code development team for remediation.

New SW SCA/CR.

The admission criteria for the updated SW SCA/CR implementation are:

- Developer has completed fixing inconsistencies and/or bugs in ED code and provided SCA/CR-SW for re-run.

- IDE code with corrected discrepancies and/or bugs sent to the head of the review team or the reviewer specified by him.

Tasks to be solved when repeating static code analysis and code review:

1. Run SW SCA again
2. Swipe SW CR again
3. The verification team leader prepares an overall report on the results of the SCA/CR-SW.

The SCA/CR-Re-SW phase ends when the following criteria are met:

- The ED code provided for the second SW SCA/CR has passed this procedure.

- no deviations and/or defects were found;

- If new inconsistencies and/or deficiencies are found, they should be sent back to the ED code development team for remediation (with subsequent resumption of SCA/CR SW) (including documented explanation of the causes of individual inconsistencies and/or deficiencies cannot be resolved). This decision will be agreed between the verification team leaders and the ED code development team.

- The results of the repeated SW SCA/CR are recorded in the general report on the results of the SW SCA/CR.

To complete SW SCA/CR you need:

- all identified inconsistencies and / or shortcomings have been eliminated or the reasons why individual inconsistencies and / or shortcomings cannot be eliminated are documented (the decision on the acceptability of such declarations is agreed between the leaders of the verification team and the ED code development team and is done together).

- The verification team leader completed and approved the overall SCA/CR-SW report.

The review of the SW SCA/CR procedure is the phase in which the causes of deficiencies identified during the SW SCA/CR are classified. This activity is an important step in preventing the appearance of such defects in future work.

The Verification Team Leader schedules a meeting following the SCA/CR-SW process. All examiners take part in this meeting.

Tasks to be solved at the meeting, taking into account the results of the SW SCA / CR procedure:

1. Select all errors from the SCA/CR SW report for discussion.
2. Determine the causes of the error.
3. Write down the results of this meeting. Recommendations may also be made to improve existing IDE development and/or code review procedures, which are then presented to the IDE code development team to further prevent bugs.

Procedure for recording SW SCA/CR results.

All measures to implement SW SCA/CR should be documented in detail and can be entered into a bug tracking system to record and control nonconformities and/or deficiencies and track the process of resolving these nonconformities and/or deficiencies. Upon completion of the SWA SC/CR, the verification team leader should prepare a report on the documentation of the results.

The SCA/CR SW report must contain:

- data about the object SW SCA/CR;

- IS attributes from SW SCA;

- description of the SW SCA/CR implementation procedure;

- Comparative analysis of versions of IDE components that fall under the SW SCA / CR procedure (prepared as an appendix to the main report).

- primary and intermediate results of SW SWA/CR, including: rule violations,
comments,
recommendations);

- the results of assessing the impact of inconsistencies and / or deficiencies identified during the implementation of the SW SCA / CR on other documents (companies) previously developed within the framework of the relevant project being considered (indicate the need to make changes) in accordance with the procedure described in the internal instructions);

- results of re-SW SCA/CR after conflict resolution (if found during SW SCA/CR)

If identified inconsistencies and/or errors are not corrected, the report must include an explanation of the reasons why the ED code maintainers remained unchanged (corrected) with respect to each such violation or comment, supported by the developer's signature.

The report must be presented in an accessible form that is understandable to professionals who have not participated in the SW SCA/CR.

Overview of the main procedures for verifying the application logic of RadiCS special platforms. An important part of instrumentation and control verification (I&CS) is the verification of a specific application logic (AL) system. This section discusses the verification tasks specific to AL for the I&CS-based RadICS platform. Let us briefly describe the subject of the test.

The RadiICS platform consists of several types of modules based on the use of FPGA chips as computing, processing and system internal control systems for each of the modules. In terms of its functionality and high-level flexibility, the RadICS platform is essentially a safety PLC, except that the internal logic is executed by the FPGA instead of the processors [4–6].

The functionality of each module is driven by a common platform logic (PL) developed using an FPGA-specific IDE and implemented with hardware description languages. Developers of the I&CS-based RadICS platform cannot influence the PL, so the PL review does not concern them. The PL of the RadiICS platform was verified by the platform vendor, RPC Radiy.

AL for I&CS is developed using a specialized IDE, the Radiy Platform Configuration Tool (RPCT). RPCT provides the means to manage large, complex I&CS projects. AL is implemented with a program-specific graphical language that is very similar to the functional block diagram language specified in IEC 61131-3 [7, 8]. Fig. 2 shows the AL fragment developed in RPCT.

The main components of AL are multifunctional application blocks (AFB). Each AFB allows the user to select and use certain functions within AL-projects: logical, mathematical, synchronization, etc. Each "graphical" AFB in RPCT is tied to a hardware-implemented AFB component in an FPGA-based logic module (LM), this component performs the specified function. When compiling, RPCT converts the graphical representation of AL into AL code, which is loaded into LM and executed at runtime.

Basically, the AL code consists of the following software instructions:

1) Reading data from allocated memory cells and transferring them to a certain AFB.

2) Executing the AFB function.

3) Record the results of the AFB function in the selected memory cells.

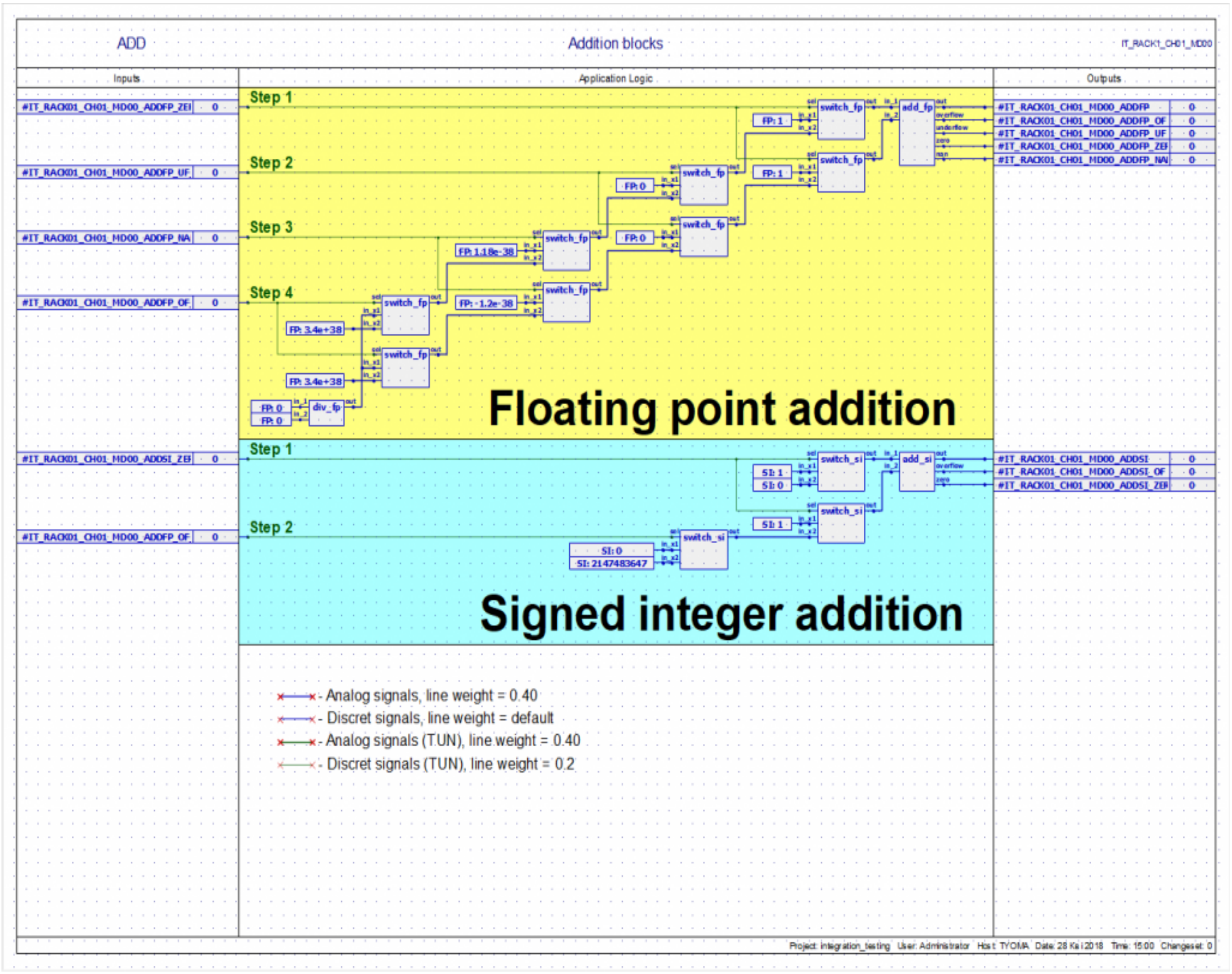

ADD
Addition blocks
IT_RACK1_CH01_MD00
Inputs
Application Logic
Outputs
Step 1
Step 2
Step 3
Step 4
#IT_RACK01_CH01_MD00_ADDFP_ZEI
#IT_RACK01_CH01_MD00_ADDFP_UF
#IT_RACK01_CH01_MD00_ADDFP_NA
#IT_RACK01_CH01_MD00_ADDFP_OF
switch_fp
add_fp
div_fp
FP: 1
FP: 0
FP: 1.18e-38
FP: -1.2e-38
FP: 3.4e+38
#IT_RACK01_CH01_MD00_ADDFP
#IT_RACK01_CH01_MD00_ADDFP_OF
#IT_RACK01_CH01_MD00_ADDFP_UF
#IT_RACK01_CH01_MD00_ADDFP_ZEI
#IT_RACK01_CH01_MD00_ADDFP_NA
Floating point addition
#IT_RACK01_CH01_MD00_ADDSI_ZEI
#IT_RACK01_CH01_MD00_ADDFP_OF
switch_si
add_si
SI: 1
SI: 0
SI: 2147483647
#IT_RACK01_CH01_MD00_ADDSI
#IT_RACK01_CH01_MD00_ADDSI_OF
#IT_RACK01_CH01_MD00_ADDSI_ZEI
Signed integer addition
- Analog signals, line weight = 0.40
- Discret signals, line weight = default
- Analog signals (TUN), line weight = 0.40
- Discret signals (TUN), line weight = 0.2
Project: integration_testing User: Administrator Host: TYOMA Date: 28 Kві 2018 Time: 15:00 Changeset: 0

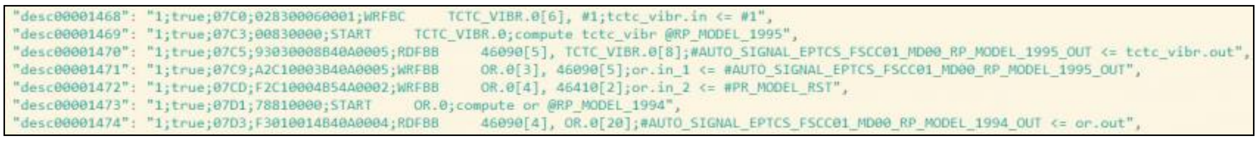
Fig. 2. Fragment of program logic, fragmentation of RPCT

In terms of data structure, AL Code is the hex code of Radiy's own assembler for FPGA-based LM.

The AL code also includes a readable part that is not loaded into LM and is for informational purposes only.

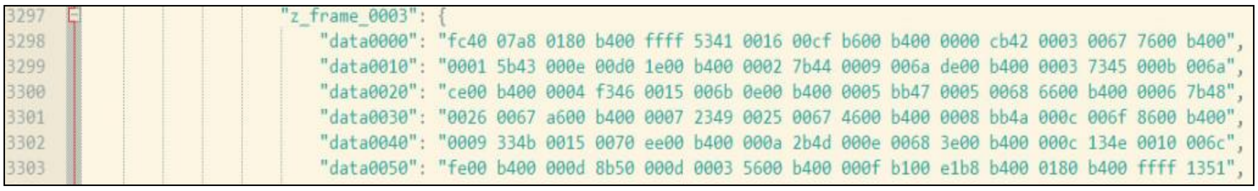
Fig. 3-4 show fragments of the logic code of the program (both parts).

The US NRC regulations, as well as the IEEE and IEC standards, provide requirements for software-related activities and supporting processes in the Safety Software Life Cycle (SLC) of nuclear power plant computerized I&CS. These standards define the different types of security-related software and the SLC requirements for each type of software.



```
"desc00001468": "1;true;07C0;028300060001;WRFBC     TCTC_VIBR.0[6], #1;tctc_vibr.in <= #1",
"desc00001469": "1;true;07C3;00830000;START     TCTC_VIBR.0;compute tctc_vibr @RP_MODEL_1995",
"desc00001470": "1;true;07C5;93030008B40A0005;RDFBB     46090[5], TCTC_VIBR.0[8];#AUTO_SIGNAL_EPTCS_FSCC01_MD00_RP_MODEL_1995_OUT <= tctc_vibr.out",
"desc00001471": "1;true;07C9;A2C10003B40A0005;WRFBB     OR.0[3], 46090[5];or.in_1 <= #AUTO_SIGNAL_EPTCS_FSCC01_MD00_RP_MODEL_1995_OUT",
"desc00001472": "1;true;07CD;F2C10004B54A0002;WRFBB     OR.0[4], 46410[2];or.in_2 <= #PR_MODEL_RST",
"desc00001473": "1;true;07D1;78810000;START     OR.0;compute or @RP_MODEL_1994",
"desc00001474": "1;true;07D3;F3010014B40A0004;RDFBB     46090[4], OR.0[20];#AUTO_SIGNAL_EPTCS_FSCC01_MD00_RP_MODEL_1994_OUT <= or.out",
```

Fig. 3. The part that is understandable for a person



```
"z_frame_0003": {
    "data0000": "fc40 07a8 0180 b400 ffff 5341 0016 00cf b600 b400 0000 cb42 0003 0067 7600 b400",
    "data0010": "0001 5b43 000e 00d0 1e00 b400 0002 7b44 0009 006a de00 b400 0003 7345 000b 006a",
    "data0020": "ce00 b400 0004 f346 0015 006b 0e00 b400 0005 bb47 0005 0068 6600 b400 0006 7b48",
    "data0030": "0026 0067 a600 b400 0007 2349 0025 0067 4600 b400 0008 bb4a 000c 006f 8600 b400",
    "data0040": "0009 334b 0015 0070 ee00 b400 000a 2b4d 000e 0068 3e00 b400 000c 134e 0010 006c",
    "data0050": "fe00 b400 000d 8b50 000d 0003 5600 b400 000f b100 e1b8 b400 0180 b400 ffff 1351",
```

Fig. 4. Hexadecimal byte code

The SLC framework for developing software using program-oriented languages is the most suitable for designing and testing AL from RPCT. The following figure is adapted from Fig. 5 in IEC 60880 and illustrates the verification and verification activities of AL (V&V) [9].

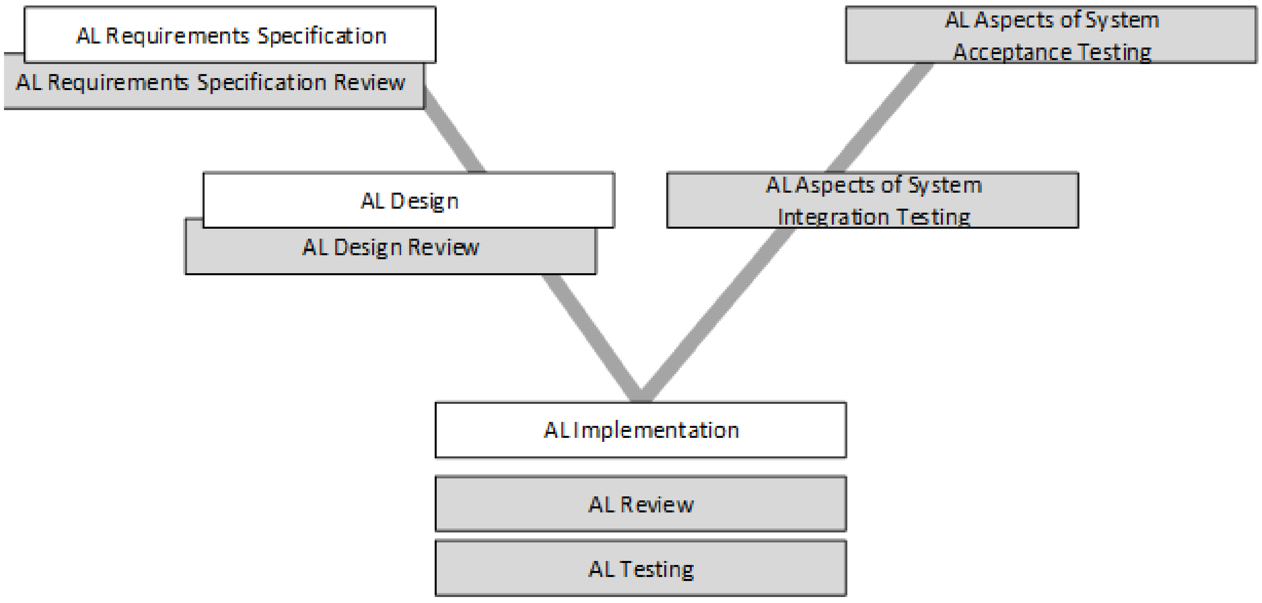

AL Requirements Specification
AL Requirements Specification Review
AL Aspects of System Acceptance Testing
AL Design
AL Design Review
AL Aspects of System Integration Testing
AL Implementation
AL Review
AL Testing

Fig. 5. Verification process and verification of program logic

As stated earlier, the AL design does not affect PL, so for a specific I&CS project, only AL needs to be tested and validated.

The top-level activities of AL V&V, such as requirements and design verification, system integration, and acceptance testing, have little to no strong specificity defined by the RadICS platform. In order to perform these top-level activities, the development organization must comply with the relevant V&V requirements for computer-based I&CS.

The AL implementation test is closely related to the language and design environment used. Therefore, this activity requires the use of specific tools and approaches.

Verification of the implementation of the QA may combine different verification tasks, such as analysis, inspections, tests or audits (depending on the requirements of the regulatory body). Viewing AL can be considered the least amount of verification effort, and can generally be sufficient for non-security applications. The combination of AL review and AL testing can be considered as a sufficient area for the most applicable security programs.

The AL Review provides an objective assessment of the AL Code. The AL review determines whether the AL code correctly implements the AL design as specified in the design documentation. The AL review should be done as a systematic check of the software.

In order to facilitate AL review, the RadICS Verification Department uses a special tool, the RPCT Outputs Verification Tool (ROVT).

The AL code consists of 2 parts: a binary part that is loaded into the FSC LM and a text part that is easy to read; both are in JSON format (see Figure 2). The binary part of the AL Code is not human readable, so does not rely on direct examination of that part of AL. ROVT was independently developed by the RadICS Verification Department to convert the binary part of the AL code into a readable visualization representing the design intent and compare both parts to confirm their equivalence.

ROVT provides the following key features:

- ROVT can be used as a mitigation to allow the AL constructor (or verifier) to verify that the compiler is correct (ie that the binaries correctly implement the graphical logic design).

- ROVT allows you to compare AL designs and highlight differences in input/output signals and AFB usage, this feature can be used to evaluate changes when planning an AL test.

- ROVT is able to generate a data path tree back from the selected output to all inputs, this function can be used for impact analysis when planning an AL test.

– ROVT performs AL static analysis. Static analysis checks are performed automatically according to user-configurable security rules.

The purpose of AL testing is to provide verification of AL requirements by performing functional software tests. AL testing is performed using instrumental behavior modeling.

Conclusions from this study and prospects for further research in this direction

The RadiICS verification department currently uses the UAL Controller Test Framework (UCTF) for AL testing purposes. UCTF contains templates for creating test inputs for AL tests, a testbed for running AL in a simulated environment, and templates for defining expected results and evaluating expected and actual results. UCTF is implemented in VHDL for ModelSim simulation. Therefore, the personnel of the verification team must have the skills and knowledge of designing and modeling VHDL.

For future use, Radiy is developing a dedicated tool in the RPCT – Simulator toolbox. The simulator executes (i.e. simulates) AL according to user-specified scenarios. The simulator can simulate user-defined parts of ALs or all ALs.

The simulator will allow the verifier to define test cases directly within the RPCT, a graphical representation of AL.

The Simulator will also support test automation in scripting languages and provide interfaces for integration with external systems (such as customer plant simulation systems). The prospect of further research in this direction is the development of software tools to improve the efficiency of using software tools for automated verification of the results of compiling software projects for critical information management systems.

References

1. Nikolayenko A.M., Minyaylo N.O. Microprocessor and software tools for automation: Training manual/ A.M. Nikolayenko, N.O Minyailo – Zaporizhzhia, ZDIA, 2011, – 444 c.
2. Striuk O., Shamanskyi V. User-Designed Application Logic Verification Approach and Tools [Electronic resource] – Access mode: https://radics.tech/user-designed-application-logic-verification-approach-and-tools/
3. IEC 61508-2010. Functional safety of electrical/ electronic/ programmable electronic safety-related systems – Part 7: Overview of techniques and measure.
4. RadICS platform [Electronic resource] – Access mode: https://radiy.com/radics/
5. Butko, I. (2021). The use of geospatial information by public authorities to support the decision making of management. Advanced Information Systems, 5(1), 39–44. https://doi.org/10.20998/2522-9052.2021.1.05
6. Davydov, V., & Hrebeniuk, D. (2020). Development the resources load variation forecasting method within cloud computing systems. Advanced Information Systems, 4(4), 128–135. https://doi.org/10.20998/2522-9052.2020.4.18
7. Khudov, H., Tahyan, K., Chepurnyi, V., Khizhnyak, I., Romanenko, K., Nevodnichii, A., & Yakovenko, O. (2020). Optimization of joint search and detection of objects in technical surveillance systems. Advanced Information Systems, 4(2), 156–162. https://doi.org/10.20998/2522-9052.2020.2.23
8. Tverytnykova, E., Demidova, Y., & Drozdova, T. (2021). Management system of occupational safety at Ukrainian enterprises: international and European dimension. Advanced Information Systems, 5(1), 45–53. https://doi.org/10.20998/2522-9052.2021.1.06
9. Software and technical complex of automatic regulation, unloading and limited power of the reactor and accelerated warning protection [Electronic resource] – Access mode: radiy.com/projectsnpp/programno-tehnichniy-kompleks-avtomatichnogo-regulyuvannya/

Received (Надійшла) 24.12.2023
Accepted for publication (Прийнята до друку) 22.02.2023

Концептуальні підходи організації процесів верифікації програмних проєктів критичних інформаційно-управляючих систем на основі вимог до їх життєвого циклу

В. В. Васюта, А. С. Данилейко, В. Б. Васюта

Анотація. Реалізація ефективної технології забезпечення необхідного рівня якості програмних систем наразі є однією з актуальних та важливих задач програмної інженерії. Особливо важливим є вирішення цієї задачі для критичних інформаційно-управляючих українських атомних електростанцій. Забезпечення необхідної якості програмних систем вимагає розуміння процесу організації верифікації програмних проектів. Метою дослідження є огляд основних концептуальних підходів організації процесів верифікації програмних проєктів критичних інформаційно-управляючих систем на основі вимог до їх життєвого циклу. Для досягнення поставленої мети авторами було розглянуто автоматизацію атомних електростанцій у вигляді програмно-технічних комплексів та загальні принципи реалізації критичних інформаційно-управляючих систем на основі програмованих логічних контролерів, проаналізовано процеси верифікації прикладної логіки користувача в даних системах, здійснено огляд основних процедур верифікації прикладної логіки спецплатформи RadiCS.

Ключові слова: програмно-технічний комплекс, критична інформаційно-управляюча система, програмований логічний контролер, верифікація.