JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 11, No. 1, March 2017, pp. 9-23

NuDE 2.0: A Formal Method-based Software Development, Verification and Safety Analysis Environment for Digital I&Cs in NPPs

Eui-Sub Kim, Dong-Ah Lee, Sejin Jung, and Junbeom Yoo*
Division of Computer Science and Engineering, Konkuk University, Seoul, Korea
atang34@konkuk.ac.kr, ldalove@konkuk.ac.kr, jsjj0728@konkuk.ac.kr, jbyoo@konkuk.ac.kr

Jong-Gyun Choi and Jang-Soo Lee
Man-Machine Interface System Team, Korea Atomic Energy Research Institute, Daejeon, Korea
choijg@kaeri.re.kr, jslee@kaeri.re.kr

Abstract
NuDE 2.0 (Nuclear Development Environment 2.0) is a formal-method-based software development, verification and safety analysis environment for safety-critical digital I&Cs implemented with programmable logic controller (PLC) and field-programmable gate array (FPGA). It simultaneously develops PLC/FPGA software implementations from one requirement/design specification and also helps most of the development, verification, and safety analysis to be performed mechanically and in sequence. The NuDE 2.0 now consists of 25 CASE tools and also includes an in-depth solution for indirect commercial off-the-shelf (COTS) software dedication of new FPGA-based digital I&Cs. We expect that the NuDE 2.0 will be widely used as a means of diversifying software design/implementation and model-based software development methodology.

Category: Embedded computing

Keywords: MBD; Formal methods; Safety analysis; PLC; FPGA; Digital I&C

I. INTRODUCTION

The programmable logic controller (PLC) [1] is an industrial computer widely used to implement safety-critical systems in digital I&Cs of nuclear power plants (NPPs). The increasing complexity of newly developed systems and maintenance costs now warrant a more powerful and cost-effective implementation platform such as the field-programmable gate array (FPGA). The nuclear industry is now eagerly researching FPGA-based digital I&Cs [2-5] to replace PLC-based systems.

However, the platform change from PLC to FPGA is not straightforward. It gives rise to a paradigm shift from CPU-based software development to gate-based hardware development. PLC software engineers should give up all experience, knowledge, and practices accumulated over decades, and start new FPGA-based hardware development from scratch. The platform change may potentially result safety-related problems. It is an urgent priority to transition safely and seamlessly to the new approach [6].

The NuDE 2.0 (Nuclear Development Environment

Open Access http://dx.doi.org/10.5626/JCSE.2017.11.1.9 http://jcse.kiise.org

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 0 A 2017; **Revised** 0 A 2017; **Accepted** 0 A 2017
*Corresponding Author

Copyright © 2017. The Korean Institute of Information Scientists and Engineers pISSN: 1976-4677 eISSN: 2093-8020

2.0), the latest version of NuDE [7-10], is a formal method-based software development, verification, and safety analysis environment for safety-critical digital I&Cs implemented with PLC and FPGA. It starts from a formal requirement specification written in NuSCR [11], and finally synthesizes C codes for PLC or Verilog/VHDL codes for FPGA, through a series of model transformations. It also supports various levels of formal verification and safety analysis to check the correctness and safety of transformed models. Verifications such as simulation, model checking, and equivalence checking are supported at each development phase, along with the provision of safety analysis such as STAMP/STPA and FTA. The 25 CASE (computer-aided software engineering) tools now mechanically and seamlessly support all model creations, transformations, verification and safety analysis.

While the NuDE (The name NuDE began to be used by [8] in 2012) was originally intended for the software development of PLC platforms, the NuDE 2.0 has been completely extended for FPGA platforms. The NuDE 2.0 makes it possible to develop software systems of PLC/FPGA platforms simultaneously from the same requirements or design specifications. We expect that the NuDE 2.0 can reduce the semantic gap between software and hardware-based developments (i.e., PLC vs. FPGA), while keeping all accumulated experience and knowledge for decades. It can also be used as a means of gaining a variety of software designs/implementations and a model-based software development methodology.

This paper explains the motivation and rationale of all techniques and supporting tools of the NuDE 2.0, and also shares an upgrade plan for the NuDE 3.0. This paper summarizes all the different case studies that we performed with several extracted reactor protection systems (RPS) examples [2, 12-14].

The organization of this paper is as follows: Section II provides overviews of the fundamental standards and guidelines for the software system development in NPPs. It also summarizes typical software development processes for PLC and FPGA-based platforms. Section III introduces the NuDE 2.0 and its supporting tools in detail. The future extension plan is also shared. Section IV briefly looks at all case studies that we have performed, and Section V compares the NuDE 2.0 with its competitors such as commercial model-based development (MBD) tools. Section VI concludes the paper.

II. THE SOFTWARE SYSTEM DEVELOPMENT IN NPPS

The software systems such as digital I&Cs in NPPs have been implemented with two platforms, PLC and FPGA. They should be developed, verified and assessed by standards and guidelines about a safety life-cycle and safety assessment, as summarized in Section II-A. The CPU-based software development process for PLC and the gate-based hardware development process for FPGA are compared in the following subsections.

A. Standards and Guidelines

Safety-critical software to implement digital I&Cs should be developed and assessed by the safety criteria of IEC and IEEE. The two organizations show different perspectives on the way to try to guarantee safety. The IEC guidelines are based on *functional safety* of IEC 61508 and try to establish a safety life-cycle in parallel with a typical software development life-cycle (SDLC, a hierarchy of regulatory guides [NUREG] and industrial codes/standards [IEEE] are applied). The plan of the standards is to realize and verify/validate safety requirements, which are developed and refined through safety/hazard/risk analysis [15] at the early phases of the safety/development life-cycle. The safety life-cycle checks iteratively whether safety requirements are implemented appropriately and sufficiently. The list below indicates IEC and IEEE standards.

- IEC 61508: Functional safety of electrical/electronic/programmable electronic (E/E/PE) safety-related systems [16]
- IEC 61513: Nuclear power plants – Instrumentation and control for systems important to safety – General requirements for systems [17]
- IEC 60880: Nuclear power plants – I&C systems important to safety – Software aspects for computer-based systems performing category A functions [18]
- IEEE Std. 603-2009: IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations [19]
- IEEE Std. 7-4.3.2-2003: IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations [20]
- IEEE Std. 1228-1994: IEEE Standard for Software Safety Plan [21]

The IEEE standards, on the other hand, require direct safety analysis at each phase of the SDLC. They suggest that all hazards (new and as well as survived) should be identified at each development phase and the hazardous conditions should be validated through various selected V&V (verification and validation) activities. Manimaran et al. [22] carried out independent V&V at each development phase, according to the IEEE standards, for a prototype fast breeder reactor.

Lee et al. [23] performed a detailed comparative analysis of the IEC and IEEE standards. Based on the experience of developing a new commercial digital I&C in Korea, the authors established a complementary relationship between the processes of development and safety analysis [24], which the NuDE 2.0 can cope with efficiently. Gabbar [25] also tried a similar approach by using process object-oriented modeling (POOM) methodology.

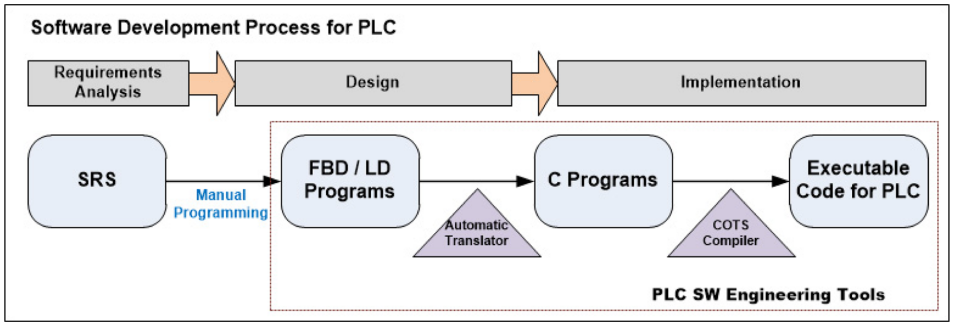

Fig. 1. A typical software development life-cycle for PLC platforms.

B. The PLC Software Development

The PLC-based digital I&Cs have a typical software development process as shown in Fig. 1. Most safety-critical systems in NPPs such as RPS and ESF-CCS (engineered safety features-components control system) have been developed with the platform. Software requirements specification (SRS) is first written in natural languages, and then the design specification is manually modeled with PLC programming languages [1] such as FBD or LD. Commercial PLC vendors provide PLC SW engineering tools (e.g., 'TriStation 1131' of Invensys for 'TriStation 1131' PLC, 'SIMATIC-Manager' of Siemens for 'SIMANTIC Controller' PLC, 'pSET' of PONU-Tech for 'POSAFE-Q' PLC [26, 27] and 'SPACE' of AREVA for 'TELEPERM XS' PLC), which mechanically translate FBD/LD programs into subsequent ANSI-C programs and executable codes for specific target PLCs. Unfortunately the commercial PLC SW engineering tools are not compatible with other tools.

Most PLC SW engineering tools also translate a high-level language such as C in order to perform verification activities such as the control flow graph (CFG)-based structural test [28] and simulation. The executable codes are too primitive to do the system/integration/unit test. Conventional software testing tools for C programs such as LDRA [29] and the one embedded in SCADE [30] can perform various testing on the C programs, while checking CFG-based structural coverages like all statements and MC/DC to assess the quality of the test cases used in [31, 32].

The problem with the approach is that the translated C program lacks enough control flows to check the CFG-based structural coverages. FBD/LD are data-flow based programming languages for PLC, and the FBD/LD programs include almost no control flows, except for a few functional blocks containing internal timers like **TOF** and **TON**. Jee et al. [33] developed 3 new data-flow based structural coverages for FBD programs and proposed a direct test of the FBD programs [34, 35].

The typical PLC software development process includes two translation/compilation steps. The translation step makes C programs with FBD programs and the compilation step makes executable codes for PLCs with C programs, which are depicted as triangles in Fig. 1. For the compilation of C programs to executable codes for PLCs, most commercial PLC SW engineering tools use commercial off-the-shelf (COTS) compilers such as 'TMS-320C55x' of Texas Instruments. The compilers were well verified and certified enough to be used without additional verification. However, the nuclear industry has not acknowledged empirically that a vendor-provided automatic compiler able to translate FBD to C is a correct and safe tool. To gain acceptance, it should be subjected to rigorous tests to demonstrate its functional safety and accuracy. There is no compiler verification technique [36, 37] for FBDs, to the best of our knowledge, and it is one of the critical obstacles for all new (so-called) FBD-to-C translators such as [27] to overcome.

C. The FPGA Software Development

An FPGA-based system has a specific feature that is classified into software as part of the development life-cycle using HDL (hardware description language), while the final chip is classified into hardware after the program is downloaded. It should be developed to meet both IEC-60880 [18] in terms of software and IEC-60987 [38] in terms of hardware criteria. Fig. 2 depicts the V-shaped life-cycle of FPGA development defined by IEC-62566 [39], consisting of software and hardware aspects. The software aspect also has a typical development life-cycle defined by NUREG/CR-7006 [40], as presented in the left-hand side of Fig. 2.

The FPGA software development (this paper uses the FPGA Software to indicate the software aspect of FPGA) is fully automated by FPGA logic synthesis tools and commercial Electronic Design Automation (EDA) tools of FPGA vendors. After programming a register-transfer level (RTL) design with HDLs, the design is mechanically transformed into a gate-level design (i.e., netlist) by synthesis software (e.g., Synopsys Synplify Pro, Precision RTL and Encounter RTL Compiler). The FPGA EDAs such as Xilinx ISE Design Suit, Altera Quartus 2, and Microsemi Libero SoC perform P&R (place & route)

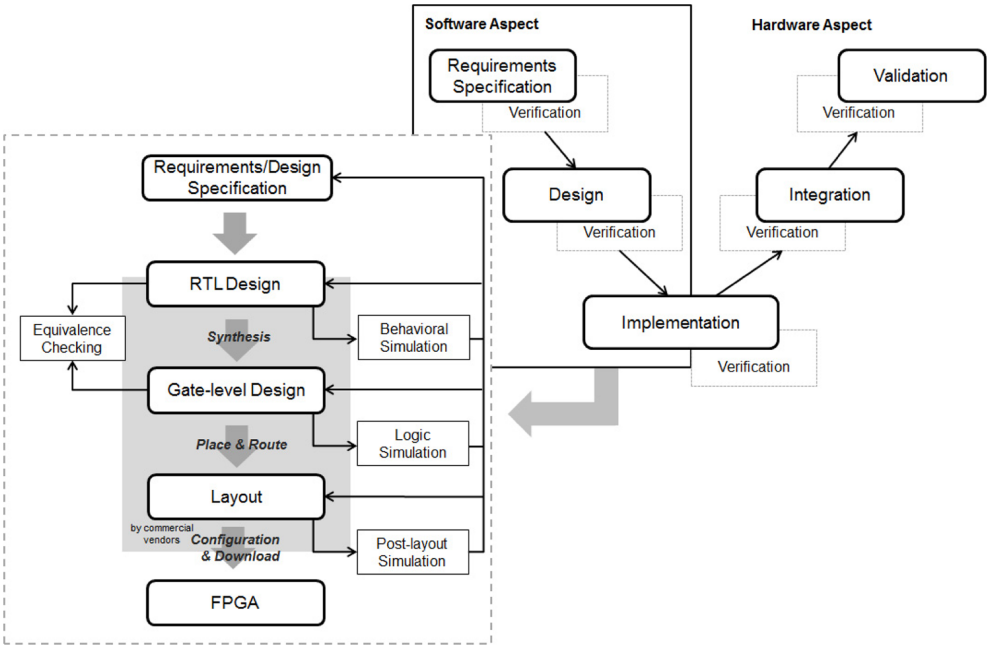

Fig. 2. A typical development life-cycle for FPGA platforms.

to physically place and map all netlist elements, and prepare a downloadable file through configuration. Since FPGA EDA tools make the synthesis process fully-automatic, software designers largely focus on HDL designs to correctly implement FPGA requirements.

At each step of the FPGA software development life-cycle, designers perform *simulation-based verification* in order to confirm that each artifact satisfies its required specification. All simulation-based verifications at each step are prepared/performed individually and repetitively by experienced engineers, and are considered to be one of key factors for efficient FPGA development.

The V&V process also includes *equivalence checking* [41, 42] and the *simulation* techniques. The equivalence checking can prove that two given designs have the same functionality, i.e., "whether they show the same behavior for all possible input sequences." For example, it can prove that an RTL design and the gate-level design synthesized from the RTL design always show the same behaviors. Most synthesis software are black-boxes of unknown quality (the FPGA industry have acknowledged them empirically as correct and safe processes and tools) and have been developed in-house by EDA company. The equivalence checking can help us ensure correct synthesis or optimization.

III. THE NUDE 2.0

The NuDE 2.0 starts from a formal requirement specification and subsequently transforms/synthesizes more concrete models across the whole SDLC, as an MBD [43] methodology for the nuclear domain. It simultaneously and seamlessly supports PLC and FPGA platforms, encompasses various formal verification and safety analysis, and the MBD-based code generation. Fig. 3 describes the whole process, techniques and CASE tools in the NuDE 2.0. The NuDE 2.0 consists of all 25 CASE tools, except for the ones marked with an asterisk (*). The following subsections explain each SDLC phase of the supporting tools.

A. The Requirements Analysis Phase

The NuDE 2.0 starts from a NuSCR specification [11] modeled in 'NuSRS 2.0' as depicted in Fig. 4. NuSCR is a data-flow based formal requirement specification language, specialized for the safety-critical systems in NPPs such as RPS and ESF-CCS. It provides 4 different notations—FOD (Function Overview Diagram), SDT (Structured Decision Table), FSM (Finite State Machine) and TTS (Timed Transition System)—to improve modeling convenience in comparison with SCR [44]. SCR provides just one notation—the decision table for all cases. NuSRS 2.0, the NuSCR modeling tool, also provides a static analyzer, Quick Checker [45], to check the syntactic completeness and consistency of NuSCR specifications. All tools underlined in Figs. 4–7 are the ones we developed.

The NuSCRtoSMV [46] translator embedded in NuSRS 2.0 generates a behaviorally-equivalent SMV input program from a NuSCR specification. After inserting CTL properties [47], which the model should satisfy, we can execute the Cadence SMV model checker [48] seamlessly and perform the model checking. The SMV model

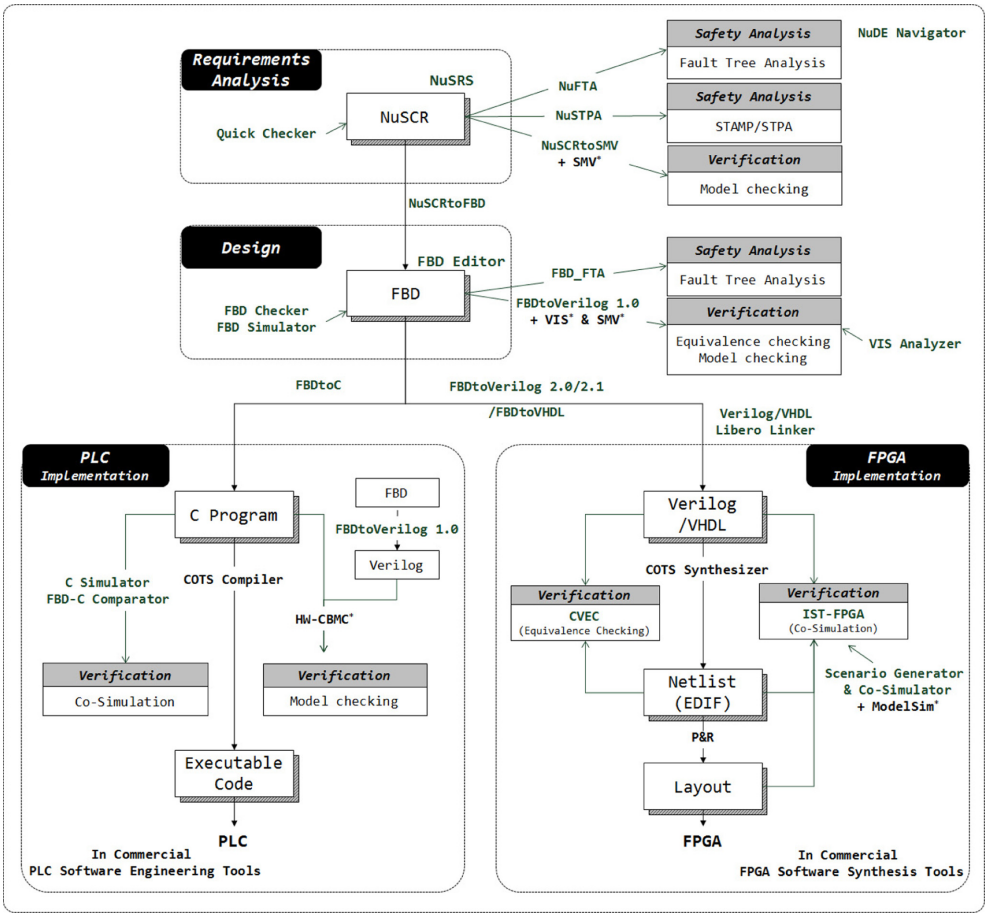

Fig. 3. An overview of the NuDE 2.0.

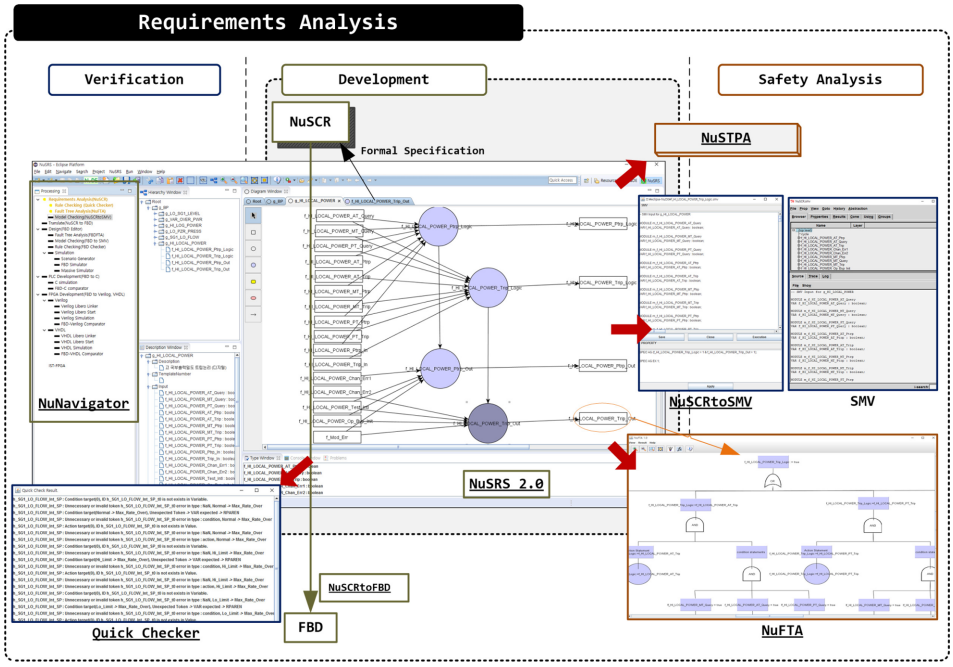

Fig. 4. The NuDE 2.0 in requirements analysis phase.

checking upon NuSCR specifications found several omitted but important assumptions [49, 50] in preliminary versions of KNICS APR-1400 RPS BP [12, 13].

The NuDE 2.0 supports two safety analysis techniques/tools in the requirements analysis phase. NuFTA [51] generates software fault trees [52] mechanically for an important output node in the NuSCR specification as shown in Fig. 4. It generates a software fault tree backwardly from the output to all inputs, and finds all combinations (i.e., conditions) of input variables, which will result in important situations such as "the shutdown signal is fired."

The NuDE 2.0 also provides the state-of-the-art safety analysis technique STAMP/STPA [53], which tries to analyze system safety from the viewpoint of system theory. It claims that "*Accidents occur when the system gets into a hazardous state, which in turn occurs because of inadequate control in the form of enforcement of the safety constraints on the system behavior.*" NuSTPA [54] helps safety engineers do the STAMP/STPA analysis on the NuSCR requirements specification. The full-scale application accompanies the extension of NuSCR and 'NuSRS 2.0', since the target of STAMP/STPA is not a SW component but a whole system consisting of many SW/HW components (e.g., RPS or NPP). The modeling target of NuSCR is now a small but critical SW component such as RPS BP.

A number of iterative modeling, verification and safety analysis produce a NuSCR specification, which fulfill the higher requirements (e.g., Functional Requirements Specification [FRS]) sufficiently and correctly. NuSCRtoFBD [55] then translates the NuSCR formal requirements specification into a behaviorally-equivalent FBD program. The FBD program plays the role of design specification for the traditional PLC-based system development. NuNavigator also shows the current phase in SDLC and helps the change into other CASE tools and SDLC phases, as shown in the upper left part of Fig. 4.

B. The Design Phase

The design phase starts from an FBD program translated from a NuSCR requirements specification by NuSCRtoFBD. FBD Editor [56] reads and displays the FBD program as depicted in Fig. 5. The FBD Editor is an independent tool from PLC vendors' SW engineering tools so that it is possible to edit FBD programs complying with the PLCopen TC6 XML scheme [57]. Commercial tools are not compatible with other editing tools for FBD programs. We can use any FBD program as a starting point, through translating it into the standard format as [58] did, if no formal specification is prepared. Programming an FBD in 'FBD Editor' from scratch is of course possible.

FBD Checker [59] checks the structure of FBD programs in accordance with several international rules and guidelines [1, 60, 61], and advises which parts may have errors or potential problems in the structure. Commercial PLC SW engineering tools perform the structural analysis well, but the exact correlation to upper rules and guidelines is not clear nor opened. FBD Checker includes a set of specific rules on the FBD structure and makes it possible to argue/acknowledge direct correlations from FBDs to rules. The example shown in the upper left part

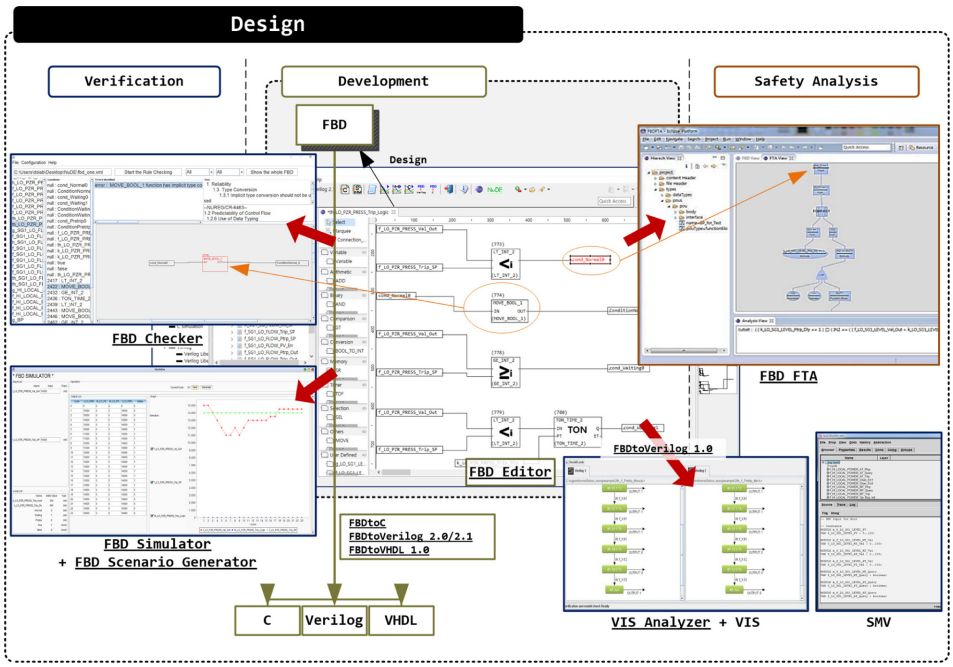

Fig. 5. The NuDE 2.0 in design phase.

of Fig. 5 advises that the function block **MOVE_BOOL_1** violates the rule, "1.3.1 Implicit type conversion should not be used." It also informs that the violated rule 1.3.1 corresponds to the higher guideline "1.2.6 Use of data typing" in NUREG/CR-6463 [61].

FBD Simulator [62] simulates any FBD program of the PLCopen TC6 standard format. It executes (i.e., simulate) the FBD program randomly or according to predefined scenarios which FBD Scenario Generator generates. Generally, PLC SW engineering tools use, store and receive each specific FBD program style or format, so that the FBD programs in different PLC engineering tools are not compatible with each other. Whereas FBD Editor and FBD Simulator follow the industrial standard, PLCopen TC6 format, the tools provide a PLC vendor-independent FBD development environment.

FBD Scenario Generator [62] generates a number of guided scenarios mechanically from an FBD program. It requests for auxiliary information on the FBD program in order to make the generated scenario meaningful ones. Initial values and a rate of change of all input variables, trip/pretrip set-points, the overall percentage of trip situations, and the number of PLC execution cycles for each scenario are requested (Now it has been customized into the features of RPS). Our previous work [63] shows how we could use the scenario generator efficiently.

The FBDtoVerilog 1.0 translator [64, 65] makes user perform formal verification using the VIS verification system [66] and the SMV model checker. As the design phase often includes hardware-dependent modifications on FBD programs, formal verifications such as model checking and equivalence checking are additional requirements.

The NuDE 2.0 also provides VIS Analyzer [67] to assist the VIS verification. Since the VIS provides no graphical interface and even requires a series of commands to do the verification, the VIS Analyzer provides GUI to analyze the verification results efficiently and also automate many kinds of the VIS verification such as model checking, equivalence checking and simulation. The screen-dump in the lower right of Fig. 5 shows two flow-charts reorganized from a text-based verification result (i.e., counter-example).

FBD FTA [68] is a tool of mechanical fault tree generation and analysis for FBD programs. It generates a software fault tree for an important output function block in the same way with NuFTA, as shown in the upper right of Fig. 5. We are now refining it to get improvement of the generation-time. Fault tree templates [69] for FBDs can also be used to do the analysis, but safety experts have to perform manual methods [70] without automatic generation.

After a number of FBD programming iterations, verification and safety analysis, the FBD program in the FBD Editor can be transformed into two different implementation codes for PLC and FPGA, simultaneously. FBDtoC [71] translates the FBD into a behaviorally-equivalent C program for PLC, while FBDtoVerilog 2.0/2.1 [72, 73] transforms it into a Verilog program for FPGA. FBDtoVHDL [74] transforms the FBD into a VHDL program, too. Additionally, the NuDE 2.0 provides an automatic linking program, Libero Linker [75], for the FPGA EDA of Actel. It reads the Verilog/VHDL program, creates a Libero project, and executes Actel Libero SoC.

C. The PLC Implementation Phase

The C programs transformed by FBDtoC [71] are then compiled into executable codes for a specific target PLC. Most commercial PLC SW engineering tools use COTS compilers, which were well verified and certified enough to be used without additional verification. However, the vendor-provided automatic translators from FBD to C, such as pSET [27] and FBDtoC, should be demonstrated to be functionally safe and accurate through rigorous tests.

FBDtoC [71] defined all FBD elements formally and proposed 1:1 translation algorithms from all FBD elements to corresponding C elements. It generates a hierarchy of ANSI-C programs, consisting of basic functions, components and a system. We acknowledge their behavioral equivalence through looking into their 1:1 correspondence between all elements. The behavioral equivalence can even be formally verified, if necessary, using the HW-CBMC model checker [76] through the verification process we proposed [77].

C Simulator and C Scenario Generator [62, 78] test (i.e., simulate/execute) the intermediate C programs. C Scenario Generator generates a number of guided simulation scenarios for C programs, trying to reflect the physical conditions for the RPS trip logics. It is similar to the FBD Scenario Generator. C Simulator executes the ANSI-C programs. These tools support the efficient system testing of PLC software.

C Simulator and C/FBD Scenario Generator are also used to demonstrate the safety and correctness of the vendor-specific (so-called) FBD-to-C translators. C/FBD Scenario Generator generates scenarios for FBD and C programs, while C Simulator and FBD Simulator executes the ANSI-C and FBD programs, respectively. FBD-C Comparator reads the sets of FBD/C programs and scenarios, executes the both sets, compares simulation results, and finally decides their behavioral equivalence, as summarized in Fig. 6.

In summary, most commercial PLC SW engineering tools read FBD programs and generate C programs and executable codes for PLCs without human intervention. We must use the commercial tools when developing FBD programs, even if the target PLC is not decided yet. On the other hand, FBDtoC can transform all FBD programs written in the PLCopen TC6 format into a set of behaviorally-equivalent ANSI-C programs. With the help of C Simulator and Scenario Generator, we can perform system tests upon the C programs. While FBDtoC provides a straightforward translation from all FBD elements into all

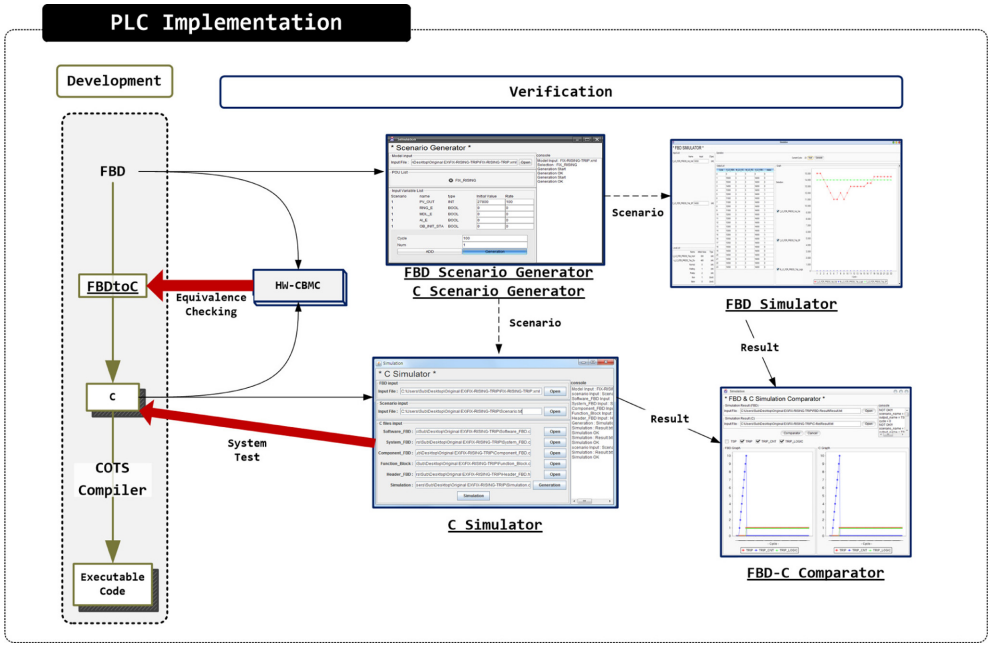

Fig. 6. The NuDE 2.0 in the PLC implementation phase.

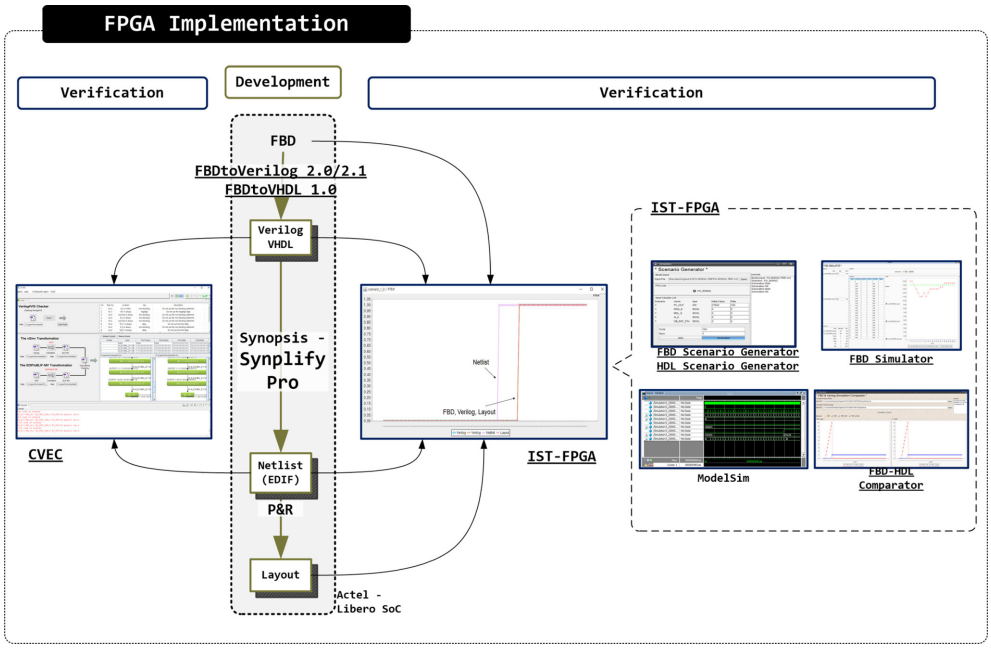

Fig. 7. The NuDE 2.0 in the FPGA implementation phase.

corresponding C elements, formal verification with HW-CBMC and the co-simulation with FBDC Comparator can be used effectively to demonstrate the safety and correctness of new FBD-to-C translators.

D. The FPGA Implementation Phase

The Verilog program translated by FBDtoVerilog 2.0/2.1 [72, 73] and the VHDL program by FBDtoVHDL 1.0 [74] are the starting point of the fully-automated FPGA synthesis procedure provided by commercial EDA tools, as shown in Fig. 7. The NuDE 2.0 can also start from the Verilog/VHDL programs programmed by software engineers from scratch. Although any commercial EDA tools can read the Verilog and VHDL programs, Synplify Pro is a specific case tool in NuDE 2.0. The CVEC and IST-FPGA only aim to verify the tool.

Nuclear regulation authorities, however, require more

considerate/rigorous demonstration of the correctness and even safety of the mechanical synthesis processes of FPGA synthesis tools, even if the FPGA industry has acknowledged them empirically as correct and safe processes and tools. We, therefore, have to get the indirect COTS SW dedication [79] upon the commercial FPGA synthesis tools. While the synthesis process can be formally verified with the compiler verification techniques [36, 37], it is hard to apply them to the works of 3rd-party developers. It must be the most critical obstacle for FPGAs to be used as a new platform of digital I&Cs. We have tried to overcome it through a safety-and-correctness demonstration technique we proposed in [80].

The proposed solution is to do the indirect demonstration [81]. For a specific program (e.g., a Verilog program), if a synthesis tool produces a program (e.g., Netlist) that shows the same behavior for all possible cases, then we can claim that the synthesis tool works correctly at least for the program. There are several commercial formal verification tools which can be used for our purpose such as FormalPro, Encounter Conformal EC, and Formality. They are, however, too case-sensitive to use naïvely, as depending upon the combination of synthesis, EDA and verification tools, as summarized in [82]. For example, we cannot use FormalPro for Actel Libero IDE with Synopsys Synplify Pro synthesizer, which is the combination of the project we worked with. The FormalPro, however, requires additional information such as register/variable matching or libraries from synthesis tools. We cannot use the tools without supporting vendors. We needed to develop a new customized solution for this combination, since the vendors cannot expect to get a lot of profit from the extension.

CVEC [82] is a VIS-based equivalence checker, customized for the combination above. It formally checks the behavioral equivalence between a Verilog program and a Netlist (i.e., EDIF) subsequently synthesized by Synopsys Synplify Pro in the Actel Libero IDE environment. If the formal verification with CVEC succeeds, we can claim that the logic synthesis from Verilog into Netlist worked correctly at least for the Verilog program used. FPGA software designers often use the simulation-based verification techniques [83-85] in order to check if high-level designs are correctly synthesized into low-level ones. At each step (i.e., RTL, gate-level and layout), designers perform three similar activities. They first develop test scenarios, simulate each target in a test bench, and finally evaluate the simulation results against specified requirements. The problem is that they should perform the verification activity at each step individually and repetitively, and it takes considerable time and cost.

IST-FPGA [63] provides an integrated software testing framework for FPGA software developments. It allows us to perform the three activities only once and in one step. For all design artifacts at every step, it generates common and meaningful test scenarios mechanically, simulates all designs simultaneously, and finally evaluates the simulation results against expected ones all together. If any one of designs shows different (i.e., incorrect) behaviors from the expected ones, IST-FPGA analyzes and compares the incorrect case in detail.

In summary, commercial FPGA EDA tools provide fully-automatic FPGA SW synthesis. Nuclear regulation authorities, however, require more considerate/rigorous demonstration of the correctness of the automatic synthesis. CVEC provides formal equivalence checking between an RTL program and a Netlist in order to demonstrate the correctness of the Synopsys Synplify Pro synthesizer in the 'Actel Libero IDE' EDA. IST-FPGA also provides an integrated software simulation (testing) framework, which can generate and execute simulation cases simultaneously for all phases of FPGA SDLC.

E. The Challenges and Future Extension Plans for NuDE 3.0

The challenges in the NuDE 2.0 is to improve the quality with regard to a traceability of the processes and to assure safety of final program. The NuDE 2.0 helps the user to smoothly use all of individual tools such as editing and translation tools. The translators in NuDE 2.0 are whole automatic system so that users do not need to consider the internal behavior. However, the whole automatic assistance may reduce a tracing ability, which is an

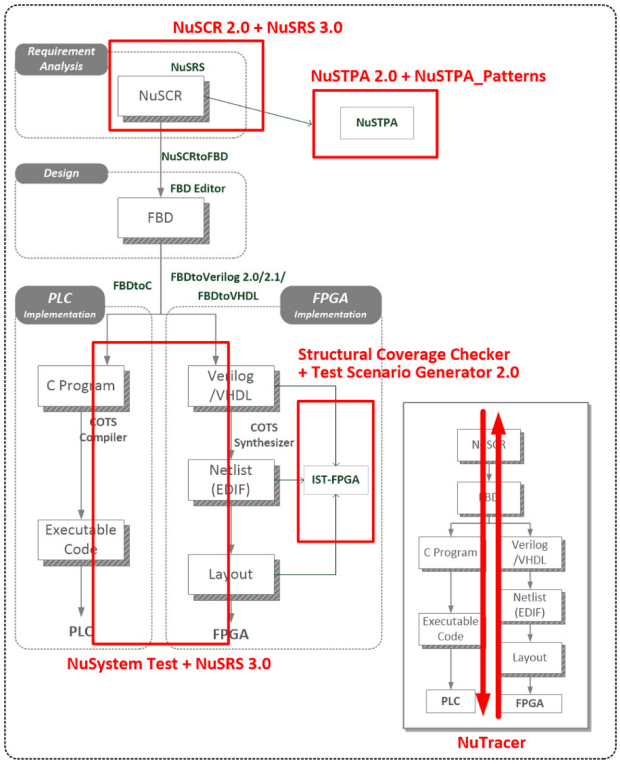

Fig. 8. A brief look at NuDE 3.0.

important factor in software development life-cycle. We need to develop a new environment in order to trace the important clues during software development life-cycle.

With regard to safety, the NuDE 2.0 only uses fault tree analysis (FTA) method, which may be lacking diversity. We also have to supplement other methods such as STAMP/STPA and Safety case to get a variety of insights.

We are now planning to extend the NuDE 2.0 to incorporate several advanced facilities. Fig. 8 shows the overall plan of the NuDE 3.0. The NuDE 3.0 will include an extended version of NuSCR formal requirements specification, which can handle not only a single SW but also a whole system consisting of many SW and HW. It will also include new structural testing coverages for RTL/gate levels, and a mechanical coverage checker for Verilog/VHDL/Netlist programs will be developed. An automatic generation of simulation test cases according to defined structural coverages will be supported, too. Forward and backward traceability analysis on the whole elements of the NuDE will be the most important contribution of the NuDE 3.0.

- NuSCR 2.0 & NuSRS 3.0: Extending NuSCR to incorporate a SW-based system as well as a single SW
- NuSTPA 2.0: The STAMP/STPA analysis on NuSCR 2.0
- NuSTPA_Patterns: STAMP/STPA patterns for digital I&Cs
- Structural Coverage Checker: Checking structural coverages of Verilog/VHDL/Netlist programs in IST-FPGA
- Scenario Generator 2.0: Generating simulation scenarios according to structural testing coverages
- NuSystem Tester: Assisting and automating the system test execution on an FPGA SW system, starting from requirements analysis phase
- NuTracer: Tracing functional requirements up to implementation codes and test cases

All elements of the NuDE 3.0 will be reimplemented with the state-of-the-art development platform, Eclipse RCP (rich client platform). The NuDE 2.0 used the Eclipse plug-in to integrate/manage 25 tools.

IV. CASE STUDY

We have performed various case studies to demonstrate the effectiveness and applicability of the NuDE 2.0. Fig. 9 summarizes all partial/full scale case studies for 25 techniques/tools of the NuDE 2.0. We have used 4 example systems.

[Example System I] starts from a NuSCR formal requirements specification [12] for a preliminary version of KNICS APR-1400 RPS BP. It consists of 8 representative shutdown logics for the RPS BP. The formal specification is translated to a behaviorally-equivalent FBD

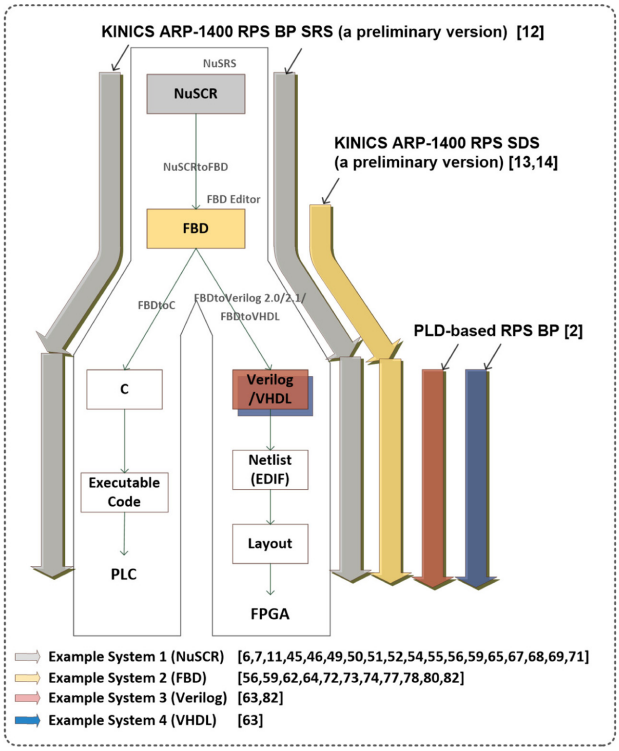

Fig. 9. All case studies for the NuDE 2.0.

program by NuSCRtoFBD, and also translated to a C program for PLC implementation by FBDtoC and a Verilog program for FPGA implementation by FBDtoVerilog 2.0. More than 20 case studies [6, 7, 49, 55, 69, 71] were performed with the Example System I.

[Example System II] starts from an FBD program [14] for the second phase of KNICS APR-1400 RPS BP [13]. It was excerpted from an almost (but, not officially final version) commercial NPP in operation and it is much more complicated and detailed than the Example System I. It consists of 18 shutdown logics of FBD programs, and FBDtoVerilog 2.0/2.1 transform them into Verilog programs. Many case studies [64, 77, 80, 82] focusing on safety/correctness demonstration of commercial FPGA synthesis tools used the Example System II.

[Example System III & IV] start from Verilog and VHDL programs, respectively, for an experimental programmable logic device (PLD)-based RPS BP [2] in Korea. They consist of 18 shutdown logics as commercial RPS BPs, but are experimental systems with fundamental functionalities. Recent case studies in CVEC [82] and IST-FPGA [63] used the Example System III and IV.

It is worthwhile to note that the NuDE 2.0 has been developed, refined and improved for more than 10 years. Now it consists of 25 tools which can seamlessly communicate/link with each other. A number of case studies with the 4 example systems have been tried to demon-

Table 1. A comparison table of commercial MBD tools

	SCADE Suite	Simulink	Rhapsody	Rose RealTime	ASCET	NuDE
SDLC						
Requirements	O	X	O	X	X	O
Design	O	O	O	O	O	O
Implementation	O	O	O	O	O	O
Safety analysis	X	X	FTA	X	X	FTA, STPA
Formal verification	O	O	O	X	O	O
Code generation	C, Ada	C, C++, ST, HDL	C, C++, Java, Ada	C, C++, Java, CORBA	C	C, FBD, Verilog, VHDL
Testing (simulation)	O	O	O	O	O	O
Last release	2014	2015	2015	2012	2016	2016
NPP application	[31, 92]	[93, 94]	[95]	X	X	Cases in Fig. 9

strate the correctness, effectiveness and applicability of the NuDE 2.0. This paper would like to settle down this version of the NuDE before proceeding to the next version.

The four case studies show that the NuDE 2.0 helps engineers develop an RPS software successfully where starting points are located. With the NuDE 2.0, engineers can considerably reduce the development time and effort by automatic translation tools. The safety of the final RPS software is also improved by various verification, validation and safety analysis tools.

V. RELATED WORK

This section briefly surveys and compares widely-used commercial MBD tools with the NuDE 2.0. Each one has unique characteristic specific to target systems and objectives, as summarized in Table 1. Applicability to NPP applications as well as support of code generation, safety analysis and formal verification are analyzed.

SCADE Suite [86] is gradually used to design critical software such as trains, cars, airplanes and power plants. It supports system modeling, simulation, formal verification and C/Ada code generation. Simulink [87] is a widely-used modeling and simulation environment, based on block diagrams for multi-domain dynamic systems. It provides various solvers for modeling and simulating dynamic (i.e., continuous) systems, and also offers tight integration with the MATLAB environment [88].

SCADE Suite [86] is gradually used to design critical software such as trains, cars, airplanes and power plants. It supports system modeling, simulation, formal verification and C/Ada code generation. Simulink [87] is a widely-used modeling and simulation environment, based on block diagrams for multi-domain dynamic systems. It provides various solvers for modeling and simulating dynamic (i.e., continuous) systems, and also offers tight integration with the MATLAB environment [88].

Rhapsody [89] is a UML-based visual modeling environment for real-time systems. It uses graphical UML models to generate application programs of C, C++, Java and Ada. Rose RealTime [90] is similar to Rhapsody. Rose RealTime does not support verification activities, but Rhapsody provides analysis to check deadlock, mutual exclusion and invariants. Rhapsody also provides a tool for modeling FTA and deriving safety-based requirements. ASCET [91] has been developed to meet embedded automotive requirements. It uses block diagrams and state machines to design and generate C code. It can import UML models and models of other suppliers such as Simulink.

In summary, we note that most MBD tools do not support safety/hazard analysis such as FTA and STAMP/STPA. FBD is only supported by the NuDE, while Simulink support an old and simple PLC programming language, ST (structured text). Most MBDs were used to develop PLC-based NPP applications [31, 92-95] only, but the NuDE can be used to develop both, PLC and FPGA-based NPP applications. Formal verification such as equivalence checking and model checking can only be supported by the NuDE.

VI. CONCLUSION AND FUTURE WORK

The NuDE 2.0 is a formal methods-based software development, verification and safety analysis environment for safety-critical digital I&Cs implemented with PLC and FPGA. It now consists of 25 tools which can communicate/link with each other. It makes possible to develop PLC/FPGA-based systems simultaneously from one requirement/design specification, and also helps most of the development, verification and safety analysis to be performed mechanically and seamlessly. A number of case studies with the 4 example systems have tried to

show the correctness, effectiveness and applicability of the NuDE 2.0. We are now working on the next version of the NuDE to efficiently support safety analysis, structural testing and traceability. We expect that the NuDE 2.0 will be widely used as a means of gaining diversity of software design/implementation as well as a model-based software development methodology. We also expect that it will be used to reduce the semantic gap between the PLC-based and FPGA-based developments.

ACKNOWLEDGMENTS

This research was supported by the Ministry of Science, ICT & Future Planning. It was also supported by a grant from the Korea Atomic Energy Research Institute, under the development of the core software technologies of the integrated development environment for FPGA-based controllers.

REFERENCES

1. *Programmable Controllers—Part 3: Programming languages*, International Electrotechnical Commission, IEC 61131-3, 1993.
2. J. G. Choi and D. Y. Lee, "Development of RPS trip logic based on PLD technology," *Nuclear Engineering and Technology*, vol. 44, no. 6, pp. 697-708, 2012.
3. J. Ranta, "The current state of FPGA technology in the nuclear domain," VTT Technical Research Centre of Finland, Espoo, Finland, 2012.
4. J. She, "Investigation on the benefits of safety margin improvement in CANDU nuclear power plant using an FPGA-based shutdown system," Ph.D. dissertation, The University of Western Ontario, Canada, 2012.
5. L. Lotjonen, "Field-programmable gate arrays in nuclear power plant safety automation," M.S. thesis, Aalto University, Espoo, Finland, 2013.
6. J. Yoo, J. H. Lee, and J. S. Lee, "A research on seamless platform change of reactor protection system from PLC to FPGA," *Nuclear Engineering and Technology*, vol. 45, no. 4, pp. 477-488, 2013.
7. J. Yoo, E. Jee, and S. Cha, "Formal modeling and verification of safety-critical software," *IEEE Software*, vol. 26, no. 3, pp. 42-49, 2009.
8. J. H. Lee and J. Yoo, "NuDE: development environment for safety-critical software of nuclear power plant," in *Transactions of the Korean Nuclear Society Spring Meeting*, 2012, pp. 1154-1155.
9. J. Yoo, E. S. Kim, D. A. Lee, J. G. Choi, Y. J. Lee, and J. S. Lee, "NuDE 2.0: a model-based software development environment for the PLC & FPGA based digital systems in nuclear power plants," in *Proceedings of 2014* 14th *International Symposium of Integrated Circuit (ISIC*), Singapore, 2014, pp. 604-607.
10. J. Yoo, E. S. Kim, D. A. Lee, and J. G. Choi, "An integrated software development framework for PLC & FPGA based digital I&Cs," in *Proceedings of International Symposium on Future I&C for Nuclear Power Plants/International Symposium on Symbiotic Nuclear Power System (ISOFIC/ISSNP)*, Jeju, Korea, 2014.
11. J. Yoo, T. Kim, S. Cha, J. S. Lee, and H. S. Son, "A formal software requirements specification method for digital nuclear plant protection systems," *Journal of Systems and Software*, vol. 74, no. 1, pp. 73-83, 2005.
12. Korea Atomic Energy Research Institute, "SRS for reactor protection system (KNICS-RPS-SRS101)," 2003.
13. Korea Atomic Energy Research Institute, "SRS for reactor protection system (KNICS-RPS-SRS221)," 2005.
14. Korea Atomic Energy Research Institute, "Software design specification for reactor protection system (KNICS-RPS-SDS231)," 2006.
15. C. A. Ericson, *Hazard Analysis Techniques for System Safety*, Hoboken, NJ: John Wiley & Sons, 2015.
16. *Functional safety of electrical/electronic/programmable electronic safety related systems*, International Electrotechnical Commission, IEC 61508, 2000.
17. *Nuclear power plants—Instrumentation and control important to safety—General requirements for systems*, International Electrotechnical Commission, IEC 61513:2011, 2011.
18. *Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions*, International Electrotechnical Commission, IEC 60880:2006, 2006.
19. *IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations*, IEEE Standard 603-2009, 2009.
20. *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*, IEEE Standard 7-4.3.2-2010, 2010.
21. *IEEE Standard for Software Safety Plans*, IEEE Standard 1228-1994, 1994.
22. M. Manimaran, A. Shanmugam, P. Parimalam, N. Murali, and S. S. Murty, "Software development methodology for computer based I&C systems of prototype fast breeder reactor," *Nuclear Engineering and Design*, vol. 292, pp. 46-56, 2015.
23. J. S. Lee, A. Lindner, J. G. Choi, H. Miedl, and K. C. Kwon, "Software safety lifecycles and the methods of a programmable electronic safety system for a nuclear power plant," in *Proceedings of International Conference on Computer Safety, Reliability, and Security*, Gdansk, Poland, 2006, pp. 85-98.
24. J. S. Lee, V. Katta, E. K. Jee, and C. Raspotnig, "Means-ends and whole-part traceability analysis of safety requirements," *Journal of Systems and Software*, vol. 83, no. 9, pp. 1612-1621, 2010.
25. H. A. Gabbar, "Integrated framework for safety control design of nuclear power plants," *Nuclear Engineering and Design*, vol. 240, no. 10, pp. 3550-3558, 2010.
26. PONU-Tech, "Nuclear plant design and repair services," 2015; http://www.ponu-tech.co.kr/.
27. S. Cho, K. Koo, B. You, T. W. Kim, T. Shim, and J. S. Lee, "Development of the loader software for PLC programming," in *Proceedings of Conference of the Institute of Electronics Engineers of Korea*, 2007, pp. 959-960.
28. M. Young, *Software Testing and Analysis: Process, Principles, and Techniques*, Hoboken, NJ: John Wiley & Sons, 2008.
29. Liverpool Data Research Associates, "LDRA tool suite," http://www.ldra.com.

30. Esterel Technologies, "SCADE - IEC 60880 Compliant," http://www.esterel-technologies.com/industries/iec-60880/.
31. J. H. Kim, D. Y. Oh, N. H. Lee, C. H. Kim, and J. H. Kim, "A nuclear safety system based on industrial computer," in *Transactions of the Korean Nuclear Society Spring Meeting*, 2011, pp. 963-964.
32. C. Park, C. Choe, and S. Jin, "An effective application process for code coverage analysis," in *Proceedings of International Symposium on Future I&C for Nuclear Power Plants/ International Symposium on Symbiotic Nuclear Power System (ISOFIC/ISSNP)*, Jeju, Korea, 2014.
33. E. Jee, J. Yoo, S. Cha, and D. Bae, "A data flow-based structural testing technique for FBD programs," *Information and Software Technology*, vol. 51, no. 7, pp. 1131-1139, 2009.
34. E. Jee, D. Shin, S. Cha, J. S. Lee, and D. H. Bae, "Automated test case generation for FBD programs implementing reactor protection system software," *Software Testing, Verification and Reliability*, vol. 24, no. 8, pp. 608-628, 2014.
35. D. Shin, E. Jee, and D. H. Bae, "Comprehensive analysis of FBD test coverage criteria using mutants," *Software & Systems Modeling*, vol. 15, no. 3, pp. 631-645, 2016.
36. T. Hoare, "The verifying compiler: a grand challenge for computing research," *Journal of the ACM*, vol. 50, no. 1, pp. 63-69, 2003.
37. X. Leroy, "Formal verification of a realistic compiler," *Communication of the ACM*, vol. 52, no. 7, pp. 107-115, 2009.
38. *Nuclear power plants—Instrumentation and control important to safety—Hardware design requirements for computer-based systems*, International Electrotechnical Commission, IEC 60987:2007, 2007.
39. *Nuclear power plants—Instrumentation and control important to safety—Development of HDL-programmed integrated circuits for systems performing category A functions*, International Electrotechnical Commission, IEC 62566:2012, 2012.
40. M. Bobrek, D. Bouldin, D. E. Holcomb, S. M. Killough, S. F. Smith, C. Ward, and R. T. Wood, "Review guidelines for field-programmable gate arrays in nuclear power plant safety systems," United States Nuclear Regulatory Commission, Rockville, MD, Report No. NUREG/CR-7006, 2010.
41. S. Y. Huang and K. T. Cheng, *Formal Equivalence Checking and Design Debugging*, Boston, MA: Kluwer Academic Publishers, 1998.
42. J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 4, pp. 401-424, 1994.
43. M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu, "Seamless model-based development: from isolated tools to integrated model engineering environments," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 526-545, 2010.
44. K. L. Heninger, "Specifying software requirements for complex systems: new techniques and their application," *IEEE Transactions on Software Engineering*, vol. 6, no. 1, pp. 2-13, 1980.
45. J. Jo, S. Yoon, and J. Yoo, "Improvement of quick checker for the verification of NuSCR," in *Proceedings of the Korea Conference on Software Engineering (KCSE 2011)*, 2011, pp. 393-400.
46. J. Cho, J. Yoo, and S. Cha, "NuEditor: a tool suite for specification and verification of NuSCR," in *International Conference on Software Engineering Research and Applications*, Heidelberg: Springer, 2004, pp. 19-28.
47. E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, Cambridge, MA: MIT Press, 1999.
48. K. McMillan, "Cadence SMV," http://www.kenmcmil.com/smv.html.
49. J. Yoo, S. Cha, C. H. Kim, and Y. Oh, "Formal software requirements specification for digital reactor protection systems," *Journal of KIISE: Software and Applications*, vol. 31, no. 6, pp. 750-759, 2004.
50. E. Jee, D. Shin, and D. H. Bae, "Analysis of model checking and testing and consideration of development direction for ensuring safety of RPS software," *Communications of the Korean Institute of Information Scientists and Engineer*, vol. 33, no. 7, pp. 15-26, 2015.
51. T. Kim, J. Yoo, and S. Cha, "A synthesis method of software fault tree from NuSCR formal specification using templates," *Journal of KIISE: Software and Applications*, vol. 32, no. 12, pp. 1178-1191, 2005.
52. S. Cha and J. Yoo, "A safety-focused verification using software fault trees," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1272-1282, 2012.
53. N. Leveson and J. Thomas, "An STPA primer," Massachusetts Institute of Technology, Cambridge, MA, 2013.
54. Y. Seo, "An extended process of STPA and implementation of an automatic assistant tool for reactor protection system software," M.S. thesis, Konkuk University, Seoul, 2016.
55. J. Yoo, S. Cha, C. H. Kim, and D. Y. Song, "Synthesis of FBD-based PLC design from NuSCR formal specification," *Reliability Engineering & System Safety*, vol. 87, no. 2, pp. 287-294, 2005.
56. D. A. Lee, E. S. Kim, Y. J. Seo, and J. Yoo, "FBDEditor: an FBD design program for developing nuclear digital I&C systems," in *Proceedings of the Korea Conference on Software Engineering (KCSE 2014)*, 2014, pp. 315-318.
57. PLCopen, "PLCopen for efficiency in automation," http://www.plcopen.org.
58. D. A. Lee and J. Yoo, "pSET2TC6: a translation tool to standardize the output format of pSET," in *Proceedings of the KIISE 2011 Fall Conference*, 2011, pp. 105-107.
59. S. Jung, J. Yoo, and J. S. Lee, "A platform-independent structural analysis on FBD programs for digital reactor protection systems," *Annals of Nuclear Energy*, vol. 103, pp. 454-469, 2017.
60. *Functional safety of electrical/electronic/programmable electronic safety related systems—Part 3: Software requirements*, International Electrotechnical Commission, IEC 61508-3:2000, 2000.
61. H. Hecht, M Hecht, S. Graff, W. Green, D. Lin, S. Koch, A. Tai, and D. Wendelboe, "Review guidelines on software languages for use in nuclear power plant safety systems," United States Nuclear Regulatory Commission, Rockville, MD, Report No. NUREG/CR-6463, 1996.
62. E. S. Kim, D. A. Lee, and J. Yoo, "The scenario generator for verifying the correctness of FBDtoVerilog Translator," in *Proceedings of the Korea Information Processing Society 2014 Spring Conference*, 2014, pp. 599-602.

63. J. Kim, E. S. Kim, J. Yoo, Y. J. Lee, and J. G. Choi, "An integrated software testing framework for FPGA-based controllers in nuclear power plants," *Nuclear Engineering and Technology*, vol. 48, no. 2, pp. 470-481, 2016.
64. J. Yoo, S. Cha, and E. Jee, "Verification of PLC programs written in FBD with VIS," *Nuclear Engineering and Technology*, vol. 41, no. 1, pp. 79-90, 2009.
65. J. Yoo, J. H. Lee, S. Jeong, and S. Cha, "FBDtoVerilog: a vendor-independent translation from FBDs into Verilog programs," in *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, Miami Beach, FL, 2011, pp. 48-51.
66. R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. T. Cheng, et al., "VIS: a system for verification and synthesis," in *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96)*, New Brunswick, NJ, 1996, pp. 428-432.
67. S. Jeong, J. Yoo, and S. Cha, "VIS analyzer: a visual assistant for VIS verification and analysis," in *Proceedings of the 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, Carmona, Spain, 2010, pp. 250-254.
68. Dependable Software Laboratory, "FBD FTA," http://dslab.konkuk.ac.kr/Nuclear-Requirement/FBD_FTA.htm.
69. Y. Oh, J. Yoo, S. Cha, and H. S. Son, "Software safety analysis of function block diagrams using fault trees," *Reliability Engineering & System Safety*, vol. 88, no. 3, pp. 215-228, 2005.
70. G. Y. Park, K. Y. Koh, E. Jee, and P. H. Seong, "Fault tree analysis of KNICS RPS software," *Nuclear Engineering and Technology*, vol. 40, no. 5, pp. 397-408, 2008.
71. J. Yoo, E. S. Kim, and J. S. Lee, "A behavior-preserving translation from FBD design to c implementation for reactor protection system software," *Nuclear Engineering and Technology*, vol. 45, no. 4, pp. 489-504, 2013.
72. D. A. Lee, E. S. Kim, and J. Yoo, "FBDtoVerilog 2.0: an automatic translation of FBD into Verilog to develop FPGA," in *Proceedings of the 5th International Conference on Information Science and Application (ICISA 2014)*, Seoul, Korea, 2014, pp. 447-450.
73. Dependable Software Laboratory, "FBDtoVerilog 2.10," http://dslab.konkuk.ac.kr/Nuclear-Design/FBDtoVerilog.htm.
74. J. Kim, E. S. Kim, J. Yoo, Y. J. Lee, and J. G. Choi, "FBDtoVHDL: an automatic translation from FBD into VHDL for FPGA development," *Journal of KIISE*, vol. 43, no. 5, pp. 569-578, 2016.
75. Y. Seo, D. A. Lee, and J. Yoo, "VerilogLinker: a tool for link IDE for FPGA controller to commercial FPGA synthesis software," in *Proceedings of the Korea Information Processing Society 2014 Spring Conference*, 2014, pp. 595-599.
76. E. Clarke and D. Kroening, "Hardware verification using ANSI-C programs as a reference," in *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2003, pp. 308-311.
77. D. A. Lee, J. Yoo, and J. S. Lee, "A systematic verification of behavioral consistency between FBD design and ANSI-C implementation using HWCBMC," *Reliability Engineering & System Safety*, vol. 120, no. 12, pp. 139-149, 2013.
78. E. S. Kim, "A technique for demonstrating correctness and safety of program translators: strategy and case study," M.S. thesis, Konkuk University, Seoul, 2015.
79. S. Jung, E. S. Kim, J. Yoo, J. Y. Kim, and J. G. Choi, "An evaluation and acceptance of COTS software for FPGA-based controllers in NPPs," *Annals of Nuclear Energy*, vol. 94, pp. 338-349, 2016.
80. E. S. Kim, J. Yoo, J. G. Choi, J. Y. Kim, and J. S. Lee, "A technique for demonstrating safety and correctness of program translators: strategy and case study," in *Proceedings of the 2nd International Workshop on Assurance Cases for Software-intensive Systems (ASSURE)*, Naples, Italy, 2014, pp. 210-215.
81. J. Yoo, E. S. Kim, and S. Jung, "Verification techniques for COTS dedication of commercial FPGA tools," in *Proceedings of the 10th International Symposium on Embedded Technology (ISET2015*), Daegu, Korea, 2015, pp. 150-151.
82. E. S. Kim, J. Yoo, and J. Y. Kim, "CVEC: a customized VIS-based equivalence checker for FPGA logic synthesis," *Science of Computer Programming*, 2016, submitted.
83. D. Kim, M. Ciesielski, and S. Yang, "A new distributed event-driven gate-level HDL simulation by accurate prediction," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2011, pp. 1-4.
84. D. Zheng, W. Yichen, and Z. Xueyi, "The methods of FPGA software verification," in *Proceedings of 2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, Shanghai, China, 2011, pp. 86-89.
85. R. E. Bryant, "A methodology for hardware verification based on logic simulation," *Journal of the ACM*, vol. 38, no. 2, pp. 299-328, 1991.
86. Esterel Technologies, "SCADE Suite," http://www.esterel-technologies.com/products/scade-suite/.
87. MathWorks, "Simulink," https://www.mathworks.com/products/simulink.html.
88. MathWorks, "MATLAB," https://www.mathworks.com/products/matlab.html.
89. IBM, "Rational Rhapsody," http://www.ibm.com.
90. IBM, "Rational Rose RealTime," http://www.ibm.com.
91. ETAS, "ASCET," http://www.etas.com.
92. P. Thevenod-Fosse, "Unit and integration testing of LUSTRE programs: a case study from the nuclear industry," Centre National de la Recherche Scientifique (CNRS), Toulouse, France, Report No. CNRS-LAAS-98078, 1998.
93. H. Gao, C. Wang, and W. Pan, "A detailed nuclear power plant model for power system analysis based on PSS/E," in *Proceedings of 2006 IEEE PES Power Systems Conference and Exposition (PSCE)*, Atlanta, GA, 2006, pp. 1582-1586.
94. S. A. M. Shirazi, "The theoretical simulation of a model by SIMULINK for surveying the work and dynamical stability of nuclear reactors cores," in *Nuclear Reactor*, Rijeka, Croatia: InTech, 2012.
95. P. Pihlanko, S. Sierla, K. Thramboulidis, and M. Viitasalo, "An industrial evaluation of SysML: the case of a nuclear automation modernization project," in *Proceedings of 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, Cagliari, Italy, 2013, pp. 1-8.

Eui-Sub Kim

Eui-Sub Kim is a PhD candidate in computer science and engineering at Konkuk University in Korea. He received his B.S. and M.S. degrees in computer science and engineering from Konkuk University in 2012 and 2015, respectively. His research interests include software engineering and formal verification.

Dong-Ah Lee

Dong-Ah Lee is a PhD candidate in the department of computer science and engineering at Konkuk University. He has received his B.S. and M.S. degrees at the same department and university in 2010 and 2012, respectively. His main research interests are in formal verification, hazard analysis, and software V&V.

Sejin Jung

Sejin Jung is a PhD candidate in the department of computer science and engineering at Konkuk university. He has received his B.S. and M.S. degrees at the same department and university in 2015 and 2016, respectively. His research interests are in software engineering and hazard/safety analysis.

Junbeom Yoo

Junbeom Yoo is an associate professor in Konkuk University's Department of Computer Science and Engineering. His research interests include software engineering, safety analysis and formal methods. Yoo has a Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology.

Jong-Gyun Choi

Jong-Gyun Choi received the B.S. degree in Nuclear Engineering from Hanyang University in 1994, M.S. and Ph.D. from KAIST in 1996 and 2001, respectively. He has been a principal researcher of Korea Atomic Energy Research Institute (KAERI) since 2001. His research interests include instrumentation and control system, reliability analysis, and safety assessment.

Jang-Soo Lee

Jang-Soo Lee is a principal research scientist at Korea Atomic Energy Research Institute (KAERI). He has received his M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1986, and Ph.D. degree from KAIST in 2002. His research interests include safety analysis of software based system, software verification and validation, embedded system testing, formal methods, digital instrumentation and control architecture.