VTT Technical Research Centre of Finland

Applicability of AADL in modelling the overall I&C architecture of a nuclear power plant

Linnosmaa, Joonas; Pakonen, Antti; Papakonstantinou, Nikolaos; Karpati, Peter

Published in:
Proceedings - IECON 2020

DOI:
10.1109/IECON43393.2020.9254226

Published: 18/10/2020

Document Version
Peer reviewed version

Applicability of AADL in modelling the overall I&C architecture of a nuclear power plant

Joonas Linnosmaa
VTT Technical Research Centre of Finland
Tampere, Finland
joonas.linnosmaa@vtt.fi
ORCID: 0000-0001-6852-6189

Antti Pakonen
VTT Technical Research Centre of Finland
Espoo, Finland
antti.pakonen@vtt.fi
ORCID: 0000-0002-6803-2303

Nikolaos Papakonstantinou
VTT Technical Research Centre of Finland
Espoo, Finland
nikolaos.papakonstantinou@vtt.fi

Peter Karpati
OECD Halden Reactor Project, Institute for Energy Technology
Halden, Norway
peter.karpati@ife.no

***Abstract* — This paper focuses on the challenges relating to the overall safety instrumentation and control (I&C) architectural design and more specifically the modelling and assessment of nuclear safety I&C systems at architectural level. We focus on the properties relating to Defence-in-Depth principle, mainly on the unwanted interactions between systems of different safety classification. This paper describes the design process of early conceptual overall safety I&C architecture from the modelling point of view and defines the requirements for a model-based approach to support the design and analysis of the design solution. The modelling language selected for the study was Architecture Analysis and Design Language (AADL), an architecture description language, which considers analysis as a goal. In this paper, we review the capabilities of the language for modelling overall safety I&C architectures and as a case study, we model a simplified example architecture of an APR-1400 nuclear power plant using standard AADL components and provide an overview of the analysis capabilities of the OSATE tool for checking Defence-in-Depth related requirements.**

Keywords — AADL, architecture description languages, safety I&C architecture, model-based systems engineering

I. Introduction

The safety instrumentation and control (I&C) systems of a nuclear power plant (NPP) have the purpose of enabling and supporting safe and reliable power generation, in other words, to keep the plant in steady and safe state during the different phases of plant operation. They work as the '*central nervous system*' of a nuclear power plant and through their various elements, sense basic parameters, integrate information and adjust plant operations as necessary. When there is a failure or an off-normal event, they need to respond reliably to ensure the efficient power production and safety. [1]. In general in this paper, when we mention I&C system, we mean safety I&C (which focuses on safety), instead of operation I&C (which enables power generation).

Modern digital instrumentation and control systems are highly interconnected by nature. However, at the same time it is required that this web of connecting systems and their functions — *architecture* — does not contain dependencies detrimental to safety. *The overall I&C architecture* is a term which includes (according to IAEA [2]) the system identification, classification and segmentation of safety systems, interfaces and functions allocated to them. In other words, we are not just interested in the specific technical details of an individual system, but in how a set of systems operates as a whole. (Conversely, the *I&C system architecture* partitions an individual system into redundant divisions, items to be included in each division, allocation of functions to those items, etc. [3]). In nuclear domain, the regulatory requirements call for successive levels of protection independent of each other, principle called Defence-in-Depth (DiD), which is highly relevant for the design of the overall I&C architecture.

For the reasons mentioned, designing (or even partly renewing) the overall nuclear I&C architecture according to all required safety principles and justifying the safety and behavior of the final solution is a challenging task. Even though in theory, the design of the overall I&C architecture can achieve total independence and separation of functional DiD levels, such a solution may not be practically reasonable in terms of layout or cost, raises questions relating to unnecessary complexity, operability and maintainability, and could therefore have a detrimental effect on safety [2]. In the end, the overall architecture is a trade-off between practicality and safety.

Designing a complex system-of-systems at architecture level requires approaches supporting the whole design lifecycle — the early work of the designer, and, in the later stages the work, the assessor. In the early architectural design, systems are seen in rather abstract level, the hardware and software suppliers and/or technologies are not yet necessary selected and the design focuses on fulfilling the requirements of DiD. Optimally, the selected design approach and tools would include support for reusability, scalability, verification and hierarchy. Model-based system engineering (MBSE) has the potential to offer many of these highly valued characteristics. Of particular interest are the architecture description languages (ADLs), which have been established for visualizing, specifying and testing system architecture solutions for mission critical systems in various domains. However, are they capable of supporting the design of the overall I&C architecture of a complex cyber physical system of systems like a nuclear power plant?

II. Background

In our previous work [4], we did an exploratory case study to research the capabilities of ADLs for modelling and analyzing a specific I&C system solution (reactor protection system of a ARP-1400 design). Architecture Analysis and Design Language (AADL) and Systems Modelling Language (SysML) were identified as the most promising candidates. We studied the general state-of-the-art around the subject and discussed the

differences of linear document-based and model-based approaches. In this paper, we are extending our work with AADL for the DiD related issues of the overall I&C architecture and study how to help solving them. There exists little literature on combining overall I&C architecture design with dedicated architecture modelling languages. Similar study for SysML has already been done [5], which discusses using SysML in an I&C system modernization project, an effort to move from not machine readable, even hand drawn, documentation to more reusable and traceable way of modelling. However, for the resulting large-scale model, the increasing complexity meant that the diagrams were no longer visually readable. Pihlanko et al. concluded that SysML has potential for early and high-level design, but struggles with detailed I&C.

Papers discussing the general analysis capabilities of AADL are relevant to this research. There is the **EMV2 Error Library** [6], which has also been annexed as a part of the AADL standard and is described to "*allow the specification of errors and faults that occur within a component or that may propagate across the architecture* [7]". Then there is **Behavior Annex** which "*brings into the overall architecture the ability to describe the behaviour of AADL components as well as their interactions"* [8]. It mainly brings in the ability to declare states to components and transitions between them by sending data or events across ports.

Our previous, more structured, literature review about AADL, identifying current, but more general, model-based methods supporting the safety and security in the early system stages can be found in [9]. For the specific topic of this paper, two additional relevant papers to AADL and SysML in nuclear architecture analysis applications were identified. Wei et al. [10] have created a Hazard Model Annex (HMA) in AADL to specify the hazard sources, hazards, hazard trigger mechanisms and mishaps. However, it focuses on embedded system level architecture and this is not what we are after. Wakandar et al. [11] model and analyze the dependability of architecture of a reactor trip system of a boiling water reactor using an AADL architecture model extended with EMV2 and probabilistic PRISM model checker.

III. Desing of Overall I&C architecture and Defence-in-Depth

"*Overall I&C architecture gives a high-level view of the individual I&C systems and how they relate to one another*" [2]. Development, or renewal, of the I&C systems of a nuclear power plant is done in phases, usually following some generally accepted life-cycle model; moving from requirements to concept design to detailed design. One of the important early design steps in the process is the design of the overall I&C architecture. According to IAEA's NP-T-2.11 [2] the goal of architectural design for the overall I&C is to establish:

- *The I&C systems comprising the overall architecture;*
- *The organization of these systems;*
- *The allocation of I&C functions to these systems;*
- *The interconnections across the I&C systems and the respective interactions allocated and prohibited;*
- *The design constraints (including prohibited interactions and behavior) allocated to the overall architecture;*
- *The definition of the boundaries among the various I&C systems.*

This is a part of top-down approach; the architecture design is usually done before moving to more detailed design of the specific I&C systems. Earlier in the development, the abstraction level of information is higher, and the granularity of information is coarser, higher-level decisions are done and bigger lines of design are decided. The further the development progresses, the more detailed the design information gets. Analysis methods and tools therefore need to be able to support constant re-evaluation of an evolving design and handle different levels of granularity. NP-T-2.11 also mentions one of the motivators of this research: "*Modern I&C systems are more interconnected and more difficult to analyze (and thus, safety assurance is more difficult than was the case for earlier generations of I&C systems).*"

The challenging part of the overall architecture design is balancing safety and feasibility. All the functional and non-functional requirements should be combined to optimize the overall system efficiency and to ensure the reliability of the safety functions in different situations. A report from IAEA (SSG-39) [12] lists important high level recommendations for the overall I&C architecture (tying it with the principles of DiD):

- *4.3 The overall I&C architecture should not compromise the concept of defence in depth and the diversity strategies of the design of the plant.*
- *4.8 The overall I&C architecture should define the concept of defence in depth and the diversity strategies to be applied within the over the overall I&C.*
- *4.9 The overall I&C architecture design also establishes the level of independence between the I&C systems that support the different levels of plant's concepts of defence in depth and diversity.*

Thus, managing the dependencies between different parts of the overall system and the connections between safety and non-safety systems is the key of ensuring the fulfilment of many requirements related to overall I&C architecture and DiD. It is highly beneficial to test and analyze the early design solutions in the architectural level to detect possible unwanted dependencies and faulty behavior as early as possible, enabling easier and cheaper modifications to the design.

Designing DiD is not easy, as summarized in [13], nuclear DiD requirements ask for the levels of defence to be as independent as practical from each other and from secondary systems. Dependencies are a major source of complexity in man-made systems and may be caused by unwanted interactions or shared resources in the architecture. In practice, there will always be some interdependencies between the levels of defence. DiD elements of redundancy, diversity and separation are used to fight against unnecessary dependencies and common cause failures, as it is required that these unintended dependencies on plant level should not be a source of vulnerability [3]. IAEA in [2] lists relevant DiD related architectural decisions needed to be made early on, such as:

- *The number of levels of defence in depth to be provided;*
- *The degree of independence required between levels;*

- *The manner in which non-classified systems will be separated from systems important to safety;*

To fulfil the all regulatory requirements and plant design constraints, methods and tools are needed. To be able to ensure that the key requirements for safety are achieved, they must be justified using a combination of deterministic and probabilistic safety analyses and engineering judgement [14]. During the design, the architecture must be continuously re-evaluated, as new details about the systems are made available [2].

IV. Modelling Requirements for the Overall I&C Architecture

We are interested in supporting the design of the overall I&C architecture using model-based methods and tools. Our focus is on architecture description languages (ADLs)—more specifically AADL. Could we use it to model and analyze the relevant design decisions?

First, we must clarify what components are needed for designing and modeling the overall I&C architecture. As explained in the previous chapter, and according to design guidelines set by the IAEA [12], the overall architectural design involves the allocation of I&C functions to I&C systems, and the interfaces between them. In addition, to conform to the DiD and safety class related requirements, we need to check the connections between DiD levels and safety classes of the systems for any unwanted dependencies across different levels of classes. In other words, we want to check that the architecture achieves sufficient separation. Later in the design, the I&C systems will have more dependencies, for example to support systems, such as power supplies and heating, ventilation and air-conditioning; however, we will leave them out of scope for this study. Therefore, the overall I&C architecture components we selected to be modelled with AADL are:

- Safety functions,
- DiD levels of the safety functions,
- Safety classes of the safety functions,
- I&C systems implementing the safety function(s),
- DiD levels of the I&C systems,
- Safety classes of the systems,
- Communication connections between systems.

The associations between the components are also shown as UML class diagram in Fig. **1**.

Next, we have listed some examples of general DiD related requirements (derived from NP-T-2.11), which would be useful to be analysed with an AADL model. To perform such analyses, there needs to be a way to model these requirements in AADL (e.g. SysML using requirements diagram [15]):

- No interfaces implementing communication from lower to higher safety class systems (unless there is accepted justification).
- The safety class of a system shall be at least as high as the safety class of a function allocated to it.
- The DiD levels should be separated, unless there is an accepted justification.

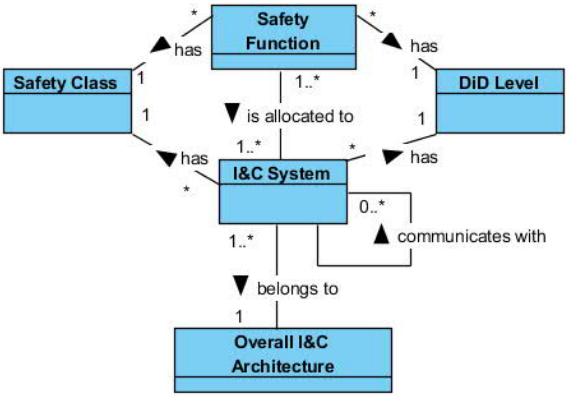

Fig. 1. Associations of overall I&C components.

To summarize the requirements, we set our goals for the modelling to be:

- Study how AADL could model and visualize the components and the architecture they form.
- Modelling of the example requirements and assessing their fulfilment.

V. AADL in Overall I&C Architecture Modelling

The architecture modelling language of particular interest to us is the AADL. We have previously in [4], [9] reviewed its applicability towards model-based specification and safety assurance of nuclear I&C systems with promising results. In this paper, we study its suitability for modelling the overall I&C architecture from the dependency and separation point of view. As explained above, the focus is on functions, systems and their connections, not on specific hardware or software components on a system architecture level. It would be advantageous if the modelling platform can be enriched with more information and details during later phases of the design.

The study of AADL in this paper is based on AADL version 2.2 published in standard AS5506C by SAE International [16]. We also used an open source tool called OSATE (Open Source AADL Tool Environment, version 2) [17] to study the AADL components. The standard gives the needed semantics and syntax to represent a system using a set of component types relating to software and hardware. The model includes the features and properties relating to those components, the connections and flows between them, and the processes driven by them. User made annexes to the original AADL standard also exist, extending AADL models with new functionalities. This paper mainly considers the components and analyses offered by the current AADL standard and OSATE, with a brief look at the analysis annexes in chapter 6.

It is crucial to understand that when the AADL standard (or AADL related literature) refers to an architecture, it usually means the system architecture of an embedded system. Compared to the overall I&C architecture we are interested in, the embedded system architecture often has more detailed safety critical requirements to be tested and verified with the AADL model. These are, for example, processor computing budgets,

end-to-end latencies and electrical power analysis. This trend is also quite implied in the summary of standard AS5506C [16]:

> "*AADL specification (model) represent a component model of a computer system runtime architecture that consist of the application software and execution platform, i.e., the computing hardware and the physical system.*"

In addition, by:

> "*AADL was developed to model embedded systems that have challenging resource (size, weight, power) constraints and strict real-time response requirements.*"

We consider those kind of models and analyses to be useful when working with the specific I&C system architectures. But, unfortunately, our interest of early overall more functional based I&C architecture specification and analysis seems to falls a bit out of scope of traditional AADL use cases offered by the standard. Nevertheless, we wanted to study how one could fulfil the requirements from Chapter 3 using general AADL components. These are our suggestions based on the AADL standard and book by Feiler and Gluch [18] on how to model overall I&C architecture components with AADL:

System: The I&C system in architectural design phase is still an abstract, but more of a physical construct, with the DiD level and safety class parameters allocated to it. Thus, the AADL generic composite component *System* is a good match. A *System instance* can later contain all the other components offered by AADL, both the execution platform and the application software (as shown in its declaration in Fig. **2**, however, for now the only interesting parameters for us are the features, subcomponents and properties). It can also be hierarchically nested with other *Systems,* which is needed, in this case for subsystems.

Category	Type	Implementation
system	Features: • port • feature group • provides subprogram access • requires subprogram access • provides subprogram group access • requires subprogram group access • provides bus access • requires bus access • provides virtual bus access • requires virtual bus access • provides data access • requires data access • feature Flow specifications: yes Modes: yes Properties: yes	Subcomponents: • data • subprogram • subprogram group • process • processor • virtual processor • memory • bus • virtual bus • device • system • abstract Subprogram calls: no Connections: yes Flows: yes Modes: yes Properties: yes

Fig. 2. Legal parameters for AADL *System* component [16].

Function: In terms of AADL components, we want a function to be something to be allocated to a *System* component. From the standard, we could not find a straight correspondence between our I&C architecture level function and a standardized AADL component (either hardware or software one). However, we decided to model the function as an AADL *Process*. It is not an ideal match, but will work as a high-level software component, for which it is possible to give all the parameters we would need (as shown in Fig. **3**, now we are interested in properties). It is also a good component to be elaborated in the future of the design process to include the realization of the software components needed to perform the safety function. Another good components choice for a safety function would have been to specify it as an *Abstract* AADL component, which can turn into any other AADL component when needed later in the process.

Category	Type	Implementation
process	Features: • port • feature group • provides data access • requires data access • provides subprogram access • requires subprogram access • provides subprogram group access • requires subprogram group access • feature Flow specifications: yes Modes: yes Properties: yes	Subcomponents: • data • subprogram • subprogram group • thread • thread group • abstract Subprogram calls: no Connections: yes Flows: yes Modes: yes Properties: yes

Fig. 3. Legal parameters for AADL *Process* component [16].

DiD level and Safety class: DiD levels and safety classes are nuclear domain related concepts, properties relating to the safety function and the I&C system. AADL has syntax for using *Properties,* which are associated values that represent attributes and characteristics of another AADL component. The standard list a default set of *Properties,* such as timing (relating to e.g. execution timing) or memory use (relating to e.g. storage). As there does not exist a straight correspondence to DiD levels or safety classes, we need to make custom *Property sets.* A *Property* needs to have a name, a type, and a value.

Connections: Connections between systems can be arranged with generic AADL *Connections*, which specify interaction between AADL components at runtime. A connection transfers state data, such as sensory data, moving from a device to another. Usually it is implemented by an AADL component *Bus,* to which the hardware systems are connected. The model can extended to with *Ports* to specify the direction of a *Flow,* which indicates whether data originates within a component, terminates within a component, or flows through a component from one of its incoming ports to one of its outgoing ports.

In Table **1**, we have summarized our example AADL counterparts for the overall I&C components introduced in the Chapter 3.

Table 1. Selected AADL counterparts for overall I&C architecture components.

Overall ICA component	AADL presentation
I&C system	System
DiD level of the I&C system	Custom property set
Safety class of the I&C system	Custom property set
Safety function	Process
DiD level of the safety function	Custom property set
Safety class of safety function	Custom property set
Connection between systems	Connections to a Bus

Requirements: requirements are rules we want our model to conform. The standard notation of AADL does not have a way to represent rules for the model itself, but uses properties of the components to set limits or resources for them to use, or give available to other components. These are non-functional requirements, such as masses, power consumptions or latencies, which are not relevant for us. To formulate our requirements we would need help of annexes, such as [19].

VI. EXAMPLE CASE: APR-1400 OVERALL I&C ARCHITECTURE

As an example to illustrate our selected AADL components in use, we use a publicly available Design Control Document (DCD) [20] about the APR-1400 nuclear reactor design from Korea Electric Power Corporation (KEPCO), submitted to the U.S Nuclear Regulatory Commission for the design certification application. However, as we are only interested in the overall I&C architecture, we do not need to go into details about individual I&C system architectures. Based on description in [20] (especially Figure 7.1-1), we developed a simplified partial high-level model of the I&C architecture of the APR-1400 design, which is shown in Fig. **4**.

As can be seen from the figure, this part overall I&C architecture includes safety and non-safety related I&C systems, as well as safety and non-safety related interfaces (generally the architectural design is done far earlier in the design process than certification, but we can this in our simple example). In the architecture, we have six systems (LDP, MTP, ITP, QIAS-N, PCS, and PPS) and the connections between them through two networks SND (safety) and DCN-I (non-safety).

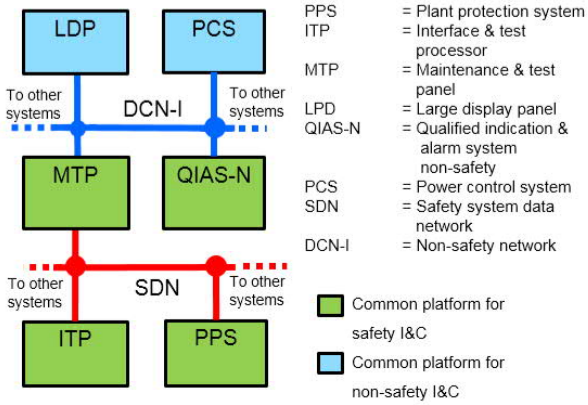

Fig. 4. Simplified partial view of APR-1400 overall I&C architecture (modified from [20]).

Based on the architecture components selected in chapter 3, more information related to the architecture is needed to develop a model like the safety functions allocated to each system and each function's and system's DiD level and safety class. As this information was not available in the public documentation, we assigned them some generic values as shown in Table 2. The estimation of these arbitrary values was based on the Finnish safety classification for nuclear systems [21], where safety class (SC) 1 is reserved for equipment in the primary circuit and systems dealing with postulated accidents belong in SC2. SC3 systems deal with anticipated operational occurrences, failure of SC2 systems, severe reactor accidents, etc.

The following paragraphs describe our modelling process and the final model of the APR-1400 example in four steps. The model was specified in AADL code editor using the OSATE tool.

Table 2. Architecture parameters for the example APR-1400 I&C systems.

System ID	Allocated function	DiD level	SC
PPS	F1	2	2
ITP	F2	2	2
MTP	F1&F2	3	2
LPD	F3	3	3
QIAS-N	F2	2	3
PCS	F4	2	Non-safety

Function ID	DiD level	SC
F1	2	2
F2	2	2
F3	3	3
F4	3	Non-safety

Step 1: We programmed an AADL custom *Property set* called 'arch', shown in Fig. **5**. This *Property set* made it possible to give a *System* and *Process* components *Properties* of safety class and DiD level as an integer value. 'SC_s' and 'DiD_s' can be given to a *System*, while 'SC_f' and 'DiD_f' mean the same, but for a *Process*.

```
property set arch is

    -- System properties
    SC_s: inherit aadlinteger  applies to (system);
    DiD_s: inherit aadlinteger  applies to (system);

    -- Function properties
    SC_f: inherit aadlinteger applies to (process);
    DiD_f: inherit aadlinteger applies to (process);

end arch;
```

Fig. 5. AADL custom *Property set* for assigning safety class and DiD level.

Step 2: These properties were then assigned to our AADL *System* implementations of the architectural systems, also specified with the required connections to the busses SND and DCN-I as seen Fig. **4**. Fig. **6** shows an AADL implementation of the PPS system. It has been assigned a system safety class of two, a DiD level of two, and a requirement for access to the SND bus. It also implements the safety function F1 as a *Process* subcomponent as specified in Table **2**.

```
system PPS
    features
        bus_access_SND: requires bus access SND.impl;
    properties
        arch::SC_s => 2;
        arch::DiD_s => 2;
end PPS;

system implementation PPS.impl
    subcomponents
        F1: process safety_function.F1;
end PPS.impl;
```

Fig. 6. AADL system component implementing an I&C system.

Step 3: The safety functions were implemented as AADL *Processes*, and they too are assigned a safety class and a DiD

level from our custom 'arch' *Property set*. Fig. **7** shows safety function F1 with safety class of two and DiD level of two.

```
process implementation safety_function.F1
    properties
        arch::SC_f => 2;
        arch::DiD_f => 2;
end safety_function.F1;
```

Fig. 7. AADL process component implementing a safety function.

Step 4: Doing the previous steps for all our systems and functions, makes it possible to combine them to form an overall architecture. Our AADL implementation of this is shown in Fig. **8**, where all the subcomponents (*Systems)* and the connections between them are realized to form a larger higher level AADL *System* component called 'overall_IC_architecture'. It contains all the I&C systems of the example, the two busses as subcomponents and it specifies the access to the required busses for the all the systems.

```
system implementation overall_IC_architecture.impl
    subcomponents
        -- systems
        MTP_sys: system MTP.impl;
        PPS_sys: system PPS.impl;
        ITP_sys: system ITP.impl;
        LPD_sys: system LPD.impl;
        QIAS_N_sys: system QIAS_N.impl;
        PCS_sys: system PCS.impl;

        -- busses
        SND_bus: bus SND.impl;
        DNC_I_bus: bus DNC_I.impl;
    connections
        -- connections to SDN
        bus_SDN_conn_MTP: bus access SND_bus <-> MTP_sys.bus_access_SND;
        bus_SDN_conn_PPS: bus access SND_bus <-> PPS_sys.bus_access_SND;
        bus_SDN_conn_ITP: bus access SND_bus <-> ITP_sys.bus_access_SND;

        -- connections to DNC-I
        bus_DNC_I_conn_MPT: bus access DNC_I_bus <-> MTP_sys.bus_access_DNC_I;
        bus_DNC_I_conn_LDP: bus access DNC_I_bus <-> LPD_sys.bus_access_DNC_I;
        bus_DNC_I_conn_QIAS_N: bus access DNC_I_bus <-> QIAS_N_sys.bus_access_DNC_I;
        bus_DNC_I_conn_PCS: bus access DNC_I_bus <-> PCS_sys.bus_access_DNC_I;

end overall_IC_architecture.impl;
```

Fig. 8. Textual implementation of the overall I&C architecture.

OSATE is also capable of visualizing the model in a diagram view. Our overall I&C architecture specified in Fig. **8** is shown in diagram view in Fig. **9**.

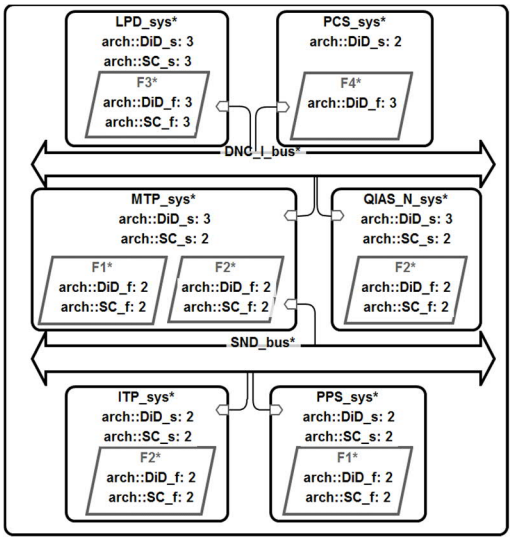

Fig. 9. Diagram view of the AADL overall I&C architecture from OSATE.

VII. AADL ANALYSES FOR OVERALL I&C ARCHITECTURE

In chapter 5, we found out that there are no standard AADL components for specifying the requirements as a part of the model. How about the available analyses themselves? Could we still perform the example checks we listed at chapter 3 on the AADL model?

The AADL standard [16] itself does not contain any information about the analyses or checks made possible with the language. However, the OSATE tool contains many checks for the models with basic AADL components, which are listed in the OSATE webpage [22]: *End-to-end Latency Analysis*, *Functional Integration Analysis*, *Port Connection Consistency Checks* (also called *Architecture Topology Analysis*), *Weight Analysis*, *Electrical Power Analysis*, *Computer Resource Budget Analysis*, *Safety Analysis*, *Structural Model Verification* and *Compositional Verification* [22]. Quite many of these are clearly related to the non-functional requirements, shortly introduced in chapter 4. No clear guidance exists on how to implement these analyses. The most relevant ones are presented below alongside a short discussion on their relevance to assessing the overall I&C architecture of a complex system.

Port Connection Consistency Check (also called **Architecture Topology Analysis):** lets users assess consistency in architecture connectivity. For example, from [22], "*ensuring that the correct types of hardware components can be interconnected, e.g., a device is connected via USB2.0, …, and ensuring that when threads with port connections are bound to different processors a hardware path via buses/networks exists between these processors.".* Although it is not directly relevant, this analysis could be used for checking existing connections with components of our interest, e.g. from lower safety class or DiD level to a higher one. It would require modification of the analysis code and implementing a connection network of ports and flows between the safety function *Processes* F1-F4 and the *Systems* as presented in Table 2.

Functional Integration Analysis: lets users *"assess consistency when components are integrated together through connections. It assures that for port connections the data types of data being communicated match that their base types, such as signed 32-bit integer, expected range of values and assumed measurement units match…*[22]". The discussion is similar to the analysis above. Based on the reference definition, it is not relevant to this case, but there is potential to be used for checking the connections between different classes and levels, if the analysis is modified and used with custom communication data types and ports.

Safety Analysis: "*supports SAE ARP4761 safety analysis for Functional Hazard Assessments (FHAs), Fault Tree Analysis (FTA), Failure Modes Effects Analysis (FMEA), Common Mode Analysis, and Reliability Block Diagrams (RBD)/Decision Diagrams (DD)* [22]*".* The ARP4761 is a guideline for Aerospace Recommended Practice published by SAE International about Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. These would be relevant in the later stages of the I&C design, but not when doing overall I&C architecture DiD analysis.

Structural Model Verification: a related toolset and an AADL annex is available in OSATE. The model verification analysis deals with model checking/requirements verification/assurance cases using Resolute language [23]. These verification processes are relevant for other parts of the analysis, such as conformity assessment, but their applicability for DiD assessment is very limited.

Compositional Verification: This analysis uses an environment called AGREE [24] to describe and analyze the requirements at different levels of abstraction for compositional verification of architectures iteratively. It tries to prove properties about one layer of the architecture using properties allocated to its subcomponents to help scale formal analysis to larger systems. Examples given in the references are for a pure software system, but the topics of scaling and architecture layers are highly relevant to modelling I&C architectures. Both of these analysis methods are important for nuclear I&C design, but at this stage too detailed for the early design phase this paper focuses on.

VIII. Results

We presented the important components needed to design and assess the overall I&C architecture of a nuclear power plant; the interconnections of systems, safety functions, DiD levels and safety classes. With the help of an overall I&C architecture diagram from APR-1400 reactor design, we discussed the potential of standard AADL components and OSATE tool for modelling and assessing the architecture for DiD related requirements.

For our first goal of the paper, we mapped the architectural components of our example case using AADL methods and created an exemplary model of the partial APR-1400 overall I&C architecture. Chapters 4, 5 and 6, demonstrated that even though AADL may not have existing components or standardized methods for modelling the overall I&C architecture, it still offers good support for modelling many of the needed architecture components. I&C systems, DiD levels, safety classes and the connections between components were implemented using basic AADL components *System, Properties*, and *Connections* respectively. However, for a safety function, no straightforward correspondence was found. We used the AADL *Process* component, but there were other options available too, such as *Abstract*. Even though we learned that the components offered by AADL are more geared towards specifying the physical decomposition of a single system, we showed that AADL and OSATE do offer ways to model and visualize the overall architecture level, too.

Related to the second goal, we did not identify a way to express the requirements using standard AADL notation. For assessing the architecture based on those requirements, no easy way was found to analyse their fulfilment. A set of promising analysis was compiled for checking the example model for unwanted connections across DiD levels or safety functions, but our model was not compatible with them, or vice versa.

IX. Discussion and Future Work

Overall, there is interest in DiD and therefore the overall I&C architecture. The Fukushima accident shed some light, internationally, on why proper DiD is important when facing threats that affect the whole plant. Modern, digital I&C systems make it easier to create emerging, unidentified, cross-discipline and inter-connections that might jeopardize DiD, underlining the need for proper methods and tools for Did modelling and assessment. Yet, formal semantics for the artefacts produced during the architectural design phase are not formally defined, which makes the modelling work.

Based on the finding of this paper, it is difficult to recommend AADL for modelling the overall I&C architecture due to the lack of analysis possibilities and standardized modelling approaches. A nuclear related custom annex with support to overall I&C components and analyses would be required to exploit the benefits of model-based approach using AADL. Currently AADL is strong when moving to lower level of hierarchies of the design, when more knowledge and decisions about the actual I&C system architecture are already available. AALD offers powerful features for modelling and analyzing the combined hardware and software layers of the system, supported with a standard. The timing and computational constrains for real time systems have increased the popularity of model-based approaches for designing such systems. AADL is a good example; its roots are heavily on software and embedded systems. Utilizing AADL in the early architecture design in nuclear domain is not straightforward, as overall I&C architectures require abstraction of components, connections and deployments. Nevertheless, AADL is a complex and constantly evolving language, and future developments might improve the situation.

In addition to improving the analysis capabilities of our model, the further work on the topic would require fitting the modelling approach and tools support better to be part of the systems engineering processes of the overall design to be truly useful for the nuclear engineers.

Acknowledgment

The Finnish Research Programme on Nuclear Power Plant Safety 2018-2022 funded this research (SAFIR2022, http://safir2022.vtt.fi). Any opinions or findings of this work are the responsibility of the authors, and do not necessarily reflect the views of the sponsors or collaborators.

References

[1] IAEA, “Core Knowledge on Instrumentation and Control Systems in Nuclear Power Plants,” *Nuclear Energy Series*, no. NP-T-3.12 Technical Report, 2011.

[2] IAEA, “Approaches for Overall Instrumentation and Control Architectures of Nuclear Power Plants,” *Nuclear Energy Series*, no. NP-T-2.11 Technical Report, 2018.

[3] T. Tommila and N. Papakonstantinou, “Challenges in Defence in Depth and I&C architectures,” *VTT Research Report*, no. VTT-R-00090-16, 2016.

[4] J. Linnosmaa, J. Valkonen, P. Karpati, A. Hauge, F. Sechi, and B. Axel Gran, “Towards model-based specification and safety assurance of nuclear I&C systems - Applicability of SysML and AADL,” *11th Nuclear Plant Instrumentation, Control, and Human-*

Machine Interface Technologies, NPIC & HMIT 2019, pp. 276–289, 2019.

[5] P. Pihlanko, S. Sierla, K. Thramboulidis, and M. Viitasalo, "An industrial evaluation of SysML: The case of a nuclear automation modernization project," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2013, pp. 1–8.

[6] S. Procter and P. Feiler, "The AADL Error Library : An Operationalized Taxonomy of System Errors," *High Integrity Language Technology (HILT) Workshop*, 2018.

[7] J. Delange and P. Feiler, "Architecture Fault Modeling with the AADL Error-Model Annex," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2014, pp. 361–368.

[8] R. B. Franca, J.-P. Bodeveix, M. Filali, J.-F. Rolland, D. Chemouil, and D. Thomas, "The AADL behaviour annex -- experiments and roadmap," in *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, 2007, pp. 377–382.

[9] A. Hauge, J. Linnosmaa, R. Fredriksen, and F. Sechi, "Safety and Security in DI&C Design – Systematic Literature Study," *HWR-1247*, 2019.

[10] X. Wei, Y. Dong, X. Li, and W. E. Wong, "Architecture-level hazard analysis using AADL," *Journal of Systems and Software*, vol. 137, pp. 580–604, Mar. 2018.

[11] A. Wakankar, A. Kabra, A. K. Bhattacharjee, and G. Karmakar, "Architectural model driven dependability analysis of computer based safety system in nuclear power plant," *Nuclear Engineering and Technology*, vol. 51, no. 2, pp. 463–478, Apr. 2019.

[12] IAEA, "Design of Instrumentation and Control Systems for Nuclear Power Plants," *Safety Standards*, no. SSG-39 Specific Safety Guide, 2016.

[13] N. Papakonstantinou, T. Tommila, B. O'Halloran, J. Alanen, and D. L. Van Bossuyt, "A model driven approach for early assessment of defense in depth capabilities of complex sociotechnical systems," in *Proceedings of the ASME Design Engineering Technical Conference*, 2017, vol. 1, p. 67257.

[14] WENRA, "Safety of new NPP designs - Study by Reactor Harmonization Working Group RHWG," 2013.

[15] M. Dos Santos Soares and J. Vrancken, "Requirements specification and modeling through SysML," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 1735–1740.

[16] SAE International, "SAE AS5506C Architecture Analysis & Design Language (AADL)." 2017.

[17] "Welcome to OSATE — OSATE 2.6.0 documentation." [Online]. Available: https://osate.org/. [Accessed: 12-Dec-2019].

[18] P. H. Feiler and D. P. Gluch, *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*. Addison-Wesley, 2012.

[19] P. Feiler, J. Delange, and L. Wrage, "A Requirement Specification Language for AADL," no. June, pp. 1--29, 2016.

[20] Korea Electric Power Corporation & Korea Hydro & Nuclear Power Co Ltd, "APR1400, Design Control Document, Tier 2, Chapter 7: Instrumentation and Controls," vol. APR1400-K-, no. September, 2018.

[21] STUK (Finnish Radiotion Safety Authority), "Classification of systems, structures and components of a nuclear facility," *YVL Guide B.2*, 2019.

[22] "About OSATE — OSATE Analysis Capabilities." [Online]. Available: https://osate.org/about-osate.html#osate-analysis-capabilities. [Accessed: 30-Dec-2019].

[23] A. Gacek, J. Backes, D. Cofer, K. Slind, and M. Whalen, "Resolute: an assurance case language for architecture models," *ACM SIGAda Ada Letters*, vol. 34, no. 3, pp. 19–28, Oct. 2014.

[24] D. Cofer, A. Gacek, S. Miller, M. W. Whalen, B. LaValley, and L. Sha, "Compositional Verification of Architectural Models," Springer, Berlin, Heidelberg, 2012, pp. 126–140.