Checking models based on an iterative co-specification process of a critical system

Fabien Bouffaron, Pascale Marangé, Gérard Morel

Checking models based on an iterative co-specification process of a critical system

Fabien Bouffaron / Pascale Marange / Gérard Morel
Université de Lorraine, CRAN, UMR 7039
CNRS, CRAN, UMR 7039, France
Vandœuvre-lès-Nancy, F-54506, France
{fabien.bouffaron, pascale.marange, gerard.morel}@univ-lorraine.fr

***Abstract*— Recurrent incident reports indicate that critical systems such as power plants exhibit unintended emergent behaviors outside of acceptable limits, despite of the increasing development of dependable automation technologies as well as of a lot of techniques related to dependability issues. Among many causes, the role of human for operating technical artefacts is of importance, but also for designing them from the early stages of specification in order to check the basic property of wholeness of any system. A mean is to ensure a continuum of compliant models to component integration through an iterative process between all the disciplines involved to engineer these systems as whole all along their life cycle. However, a human-based process to check the "right-system requirements-right" remains not fully adequate at the scale of real systems engineering projects and in any case to critical issues. This paper explores the formal checking compliance of architecting models with dependability requirements. These models are refined iteratively by specialist and specialty engineers interoperating with a system engineer through a co-specification process on a particular case study of a critical power-plant sub-system.**

Keywords—model based systems engineering; co-specification; verification/validation, model-checking; dependability requirements;

I. INTRODUCTION

Because of the resilient nature of human operators to face unintended emergent behaviors [1], a generation of power plant remains largely under human control. They are assisted in their work by automation technologies that provide the least important part of the overall control information. Control room operators and field operators interoperate together to perform documented procedures in order to align a lot of devices only partly automated to process energy flows. The increasing readiness level of automation technologies embedding somewhat of a technical intelligence is expected to enlarge the part of digital control information in order to offer new services to field operators.

This last objective is addressed by the R&D programme of the CONNEXION[1] cluster for the next generation of power plant as well as for the revamping of current ones. Its purpose is to design innovative architectures compliant with the main advances in component solutions for distributed industrial automation systems. To meet the main objective of increasing the dependability integrity levels of the overall process, the systems engineering (SE) framework is a model-based integrative approach. Its aim is to check how the system parts, and consequently the related engineering ones, interoperate such that they reproduce or transform a whole. The expected result, "in contradistinction with traditional control engineering approaches focusing first on separate parts"[2], is to check the "right-system requirements-right" between all the stakeholders involved throughout a power plant life cycle, including the enabling systems, and especially the automation system engineering. The use of models for verification and validation of dependable requirements to component solution integration is compliant with the last recommended best practices in industry [2].

The role of the system architect beyond technical aspect remains challenging and issues presented in this paper and others related ones [3], [4] are in line with those of this R&D programme. Section 2 contextualizes our CISPI-AFW case study reflecting on a lab-platform some critical features of power plant automation. It also presents the co-specification process used to engineer this system. Section 3 details a particular co-specification scenario to design a logical architecture in order to highlight the set of models refined iteratively by specialist and specialty engineers interoperating with a system engineer. Section 4 formally checks the consistency of the overall models with dependability system requirements at different stages of the studied co-specification process. To conclude, section 5 highlights on going works related to others engineering domains involved in a broader co-specification process for critical power plant control. Moreover, some perspectives are drawn on how to combine model-checking and co-simulation techniques for verification and validation issues.

[1] https://www.cluster-connexion.fr/

[2] http://www.collegepublications.co.uk/systems/syt/

II. CO-SPECIFICATION PROCESS OF THE STUDIED SYSTEM

Our case study reflects, on the one hand, some features of an auxiliary feedwater (AFW) system in emergencies. It is historically recognized as a critical sub-system [5] [6] of some power-plant generations. AFW aim is to maintain an adequate water level in the steam generator (SG) by providing feedwater to the latter, during normal operations (startup and shutdown evolutions) as well as for emergency issues (Fig.1).

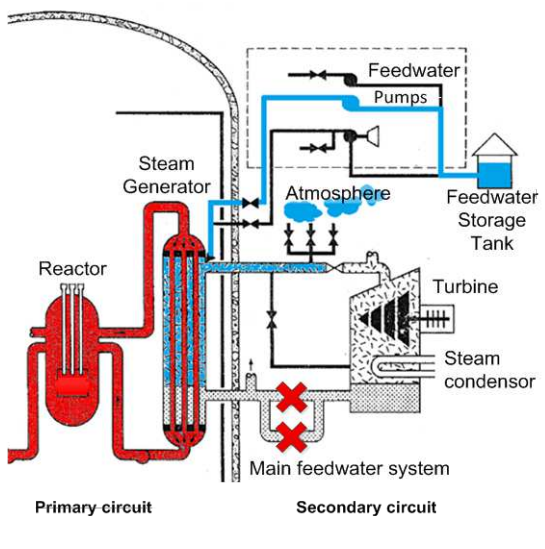

Fig. 1. AFW sub-system in emergencies (adapted from [7])

Then, our co-specification process reflects, on the other hand, a set of engineering domains involved in the automation design of such AFW sub-system. Systems engineering play an integrative role as a concurrent, iterative and recursive problem-solving process when architecting a system solution through feedbacks (Fig.2).

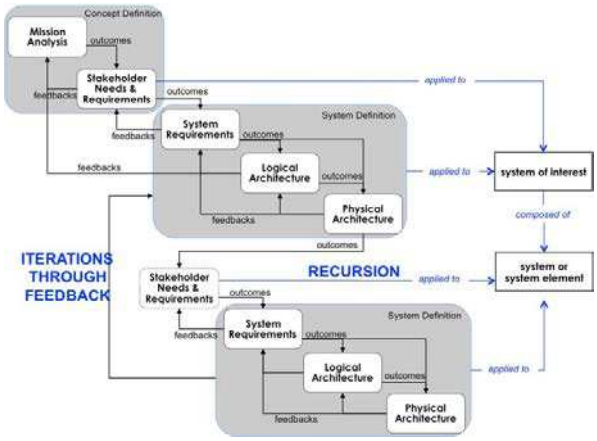

Fig. 2. Systems engineering and management paradigm [8] [9]

A. Studied Critical Sub-System

Our CISPI lab platform[3] is composed of three tanks, one of which may be associated with the feedwater storage tank and another with a steam generator. SG tank leakage output aims to simulate the flow of steam released relaxation into the atmosphere. These two tanks are interconnected by two redundant lines consisting of a manual valve, a flow meter and a control valve (Fig.3).

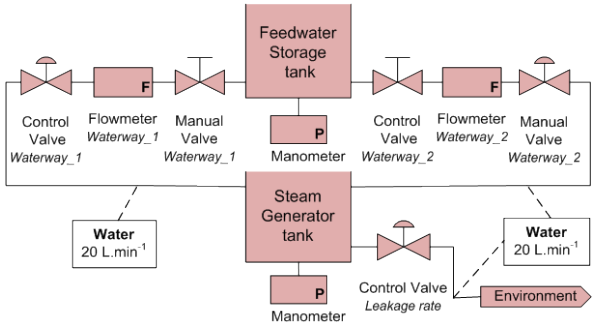

Fig. 3. Piping and instrumentation diagram of CISPI AFW-like process

The architectural concept is to distribute around a fieldbus a set of "intelligent actuation and measurement systems" (IAMS) within an integrated system of control, maintenance and technical management (CMMS), including control by human operators [10]. This paper highlights some dependability issues related to the preliminary logical definition of this architecture by several specialist and specialty engineering. This step is before the allocation of the specified function blocks to human operators and/or to automation components.

B. Studied Co-Specification Process

We revisited in previous works [3] the nature of these SE feedbacks focusing mainly on the System Breakdown Structure. (Fig.2). Following and extending the problem-frame approach for software engineering [11] related to the meaning of requirements specification [12], we argue that the basic driver of these feedbacks is first the quest of knowledge by the system architect from specialist engineering domains at each level, which is then re-iterated through the system breakdown structure.

Let's consider such a knowledge-based iteration[4] between the problem-space (PS_{SE}) of the SE domain and a solution-space (SS_{AE}) of the Automation Engineering domain. We argue that the added knowledge (K_SS_{AE}) from the AE solution-space (1) contributes iteratively to achieve the satisfaction of an automation (software-based) system specification

[3] http://safetech.cran.univ-lorraine.fr/

[4] "K_PS_{iE}" and "K_SS_{jE}" notations correspond respectively to the knowledge of problem space$_i$ (PS_{iE}) and solution space$_j$ (SS_{jE}) for the next sections

(S_{AE}) from system requirements (S_{SE}) described by an SE problem-space (2) according to entailments:

$$K\_SS_{AE}, S_{SE} \vdash S_{AE} \quad (1)$$

$$K\_PS_{SE}, S_{AE} \vdash S_{SE} \quad (2)$$

At the scale of an overall SE process restricted to our case-study, we generalize this iterative co-specification process between all the problem-space/solution-space of the engineering domains ($PS_{OE} \wedge SS_{OE}$, $PS_{SE} \wedge SS_{SE}$, $PS_{PE} \wedge SS_{PE}$, $PS_{AE} \wedge SS_{AE}$ $PS_{DE} \wedge SS_{DE}$) interoperating to satisfy the originating operational entailment:

$$W_{OP}, S_{SE} \vdash R_{OE} \quad (1)$$

where W_{OP} represents the operational environment knowledge of the target system and R_{OE} is a set of requirements described by the related operational engineering problem space (PS_{OE}).

Note that each specialist specification may be verified, for example by execution of models, within its own solution-space but must be validated by the related problem-space. In order to improve a lean management of the overall engineering workflow, we suggest distributing this co-specification process around a bus (Fig.4).

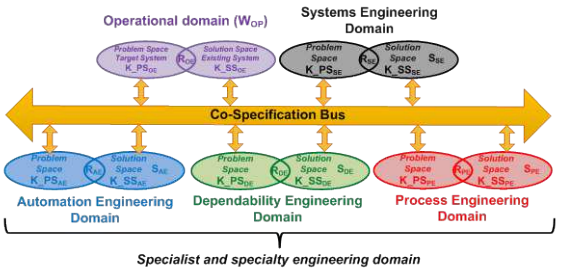

Fig. 4. Distributed co-specification process

A main interest is that engineers from various domains can use their own tools, methods, technics to interoperate with each other by the mean of a common language, such as the de-facto standardized SysML [13] in our case-study. Indeed, maturity of digital technologies for modeling and simulation allows considering the exchange and model co-execution through standardized interfaces for connecting different simulation environments. The "Functional Mockup Interface" [14] defined by the European ITEA2 Modelisar project is a promising candidate to become the industry standard for co-simulation models and cross-company collaboration [15]. The validation of a co-specification by co-execution of models can replace current technics of code generation and integration that are dependent on interoperability levels between tools. Thus, in our R&D framework, various domains and their respective tools (process engineering: *Dymola*®, automation engineering: *Matlab*®*/Simulink*®, system engineering: *IBM Rational*® *Rhapsody*) interoperate through a co-simulation bus[5].

Section III, describes a co-specification scenario between SE problem space (PS_{SE}) and specialist engineers who prescribe specification models (S_{model}) from the system specification (S_{SE}). As addressed in section IV, a special attention should be paid to the transversal dependability engineering solution space (SS_{DE}) to check more formally theses specification models satisfying:

$$K\_PS_{SE}, S_{model}, K\_SS_{DE} \vdash S_{SE}. \quad (2)$$

III. LOGICAL ARCHITECTURE CO-SPECIFICATION

We focus our scenario on the study of systems engineering activities during the co-specification of logical architecture. At this point, we consider the following set of system requirements (S_{SE}) satisfying the set of stakeholder requirements (R_{OE}) during an accident situation (W_{OP}) (Fig. 1) according to the entailment (3):

TABLE I. DRIVING CISPI-AFW SYSTEM REQUIREMENTS (S_{SE})

Stereotype	ID	Text
Functional	S_{SE1}	*The water level shall be maintained at 50 cm in the steam generator tank*
Functional	S_{SE2}	*The emergency water flow shall be 20 L.min^{-1}*
Dependability	S_{SE3}	*The feedwater system shall satisfy single failure criterion*

System engineer has to prescribe a logical architecture (LA_{SE}) requiring knowledge of several specialist solution spaces (process and automation engineering) in order to meet system requirements.

A. Specification of the AFW process

A first iteration is to specify the physical process to be controlled. From this task, there is an emergent problem about the specification of a process observing the laws of physics. Indeed, system engineer does not have the expertise to perform this task. In this sense, system engineering problem space formalizes this problem into requirements. These are then broadcast on the co-specification bus, pending a specialist engineer can address it. Process engineering solution space (SS_{PE}) being able to meet this problem (S_{SE}) specifies CISPI-AFW process (S_{PE}) according to:

$$K\_SS_{PE}, S_{SE} \vdash S_{PE} \quad (3)$$

[5] http://site.cosimate.com/

S_{PE} prescribes a process with two alignment functions (*Water_Supply*) providing transport of water between two storage functions (*Water_Storage*). So that the process engineer can interact with the system engineer, it is important that both share a common representation of the solution in a common language. In this sense, the process engineer has to externalize his specification (S_{PE}) by transcribing it in SysML language (red in Fig.5 and Fig.7). Once done, process solution space engineer prescribes S_{PE} to SE problem space for validation. CISPI-AFW being a critical system, systems engineer must formally validate at least that S_{PE} is compliant with dependability requirement S_{SE3}. However, systems engineering domain does not necessarily have the skills to formally perform this task. In this sense, a second iteration is drawn on the expertise of dependability engineering solution space to formally validate S_{PE}. This validation task is described in the next section.

B. Specification of the AFW control functions

Once the CISPI-AFW process validated, it remains to specify the control thereof. As with process engineering domain, control functions specification requires a third iteration between SE problem space and automation engineering solution space (SS_{AE}). As addressed by [14], automation as a specialist engineering has to broaden its traditional disciplinary domain in order to prescribe a control functions specification (S_{AE}) compliant with both process specification (S_{PE}) and system requirements (S_{SE}) according to entailment:

$$K\_SS_{AE}, S_{SE}, S_{PE} \vdash S_{AE} \quad (4)$$

This specification S_{AE} consists in the definition of two control functions (*Control_Water_Height*) and one observation function (*Observation_Height*) (blue in Fig. 5). In the same way as for the validation of S_{PE}, systems engineer draws again on the expertise of DE solution space to formally validate S_{AE}. This validation task is also described in the next section.

C. Specification of the AFW logical architecture

In this stage, the CISPI-AFW process specification (S_{PE}) and the control functions specification (S_{AE}) are validated by systems engineering domain. Taking into account dependability engineering specification S_{DE} ("*adding an observation function"* see next section), SE problem space (PS_{SE}) must logically architecting process and control components in order to provide a logical architecture of CISPI-AFW system (LA_{SE}) according to the entailment:

$$K\_PS_{SE}, S_{PE}, S_{AE}, S_{DE} \vdash LA_{SE} \quad (5)$$

This task requires to compromise between specialist and specialty specifications in order to prescribe LA_{SE}. Indeed, systems engineering solutions has to check that the resulting architecture of logical architectural design process is compliant with systems requirements (S_{SE}). Thus, a last iteration with dependability engineering solution space is required to verify the specified architecture (LA_{SE}).

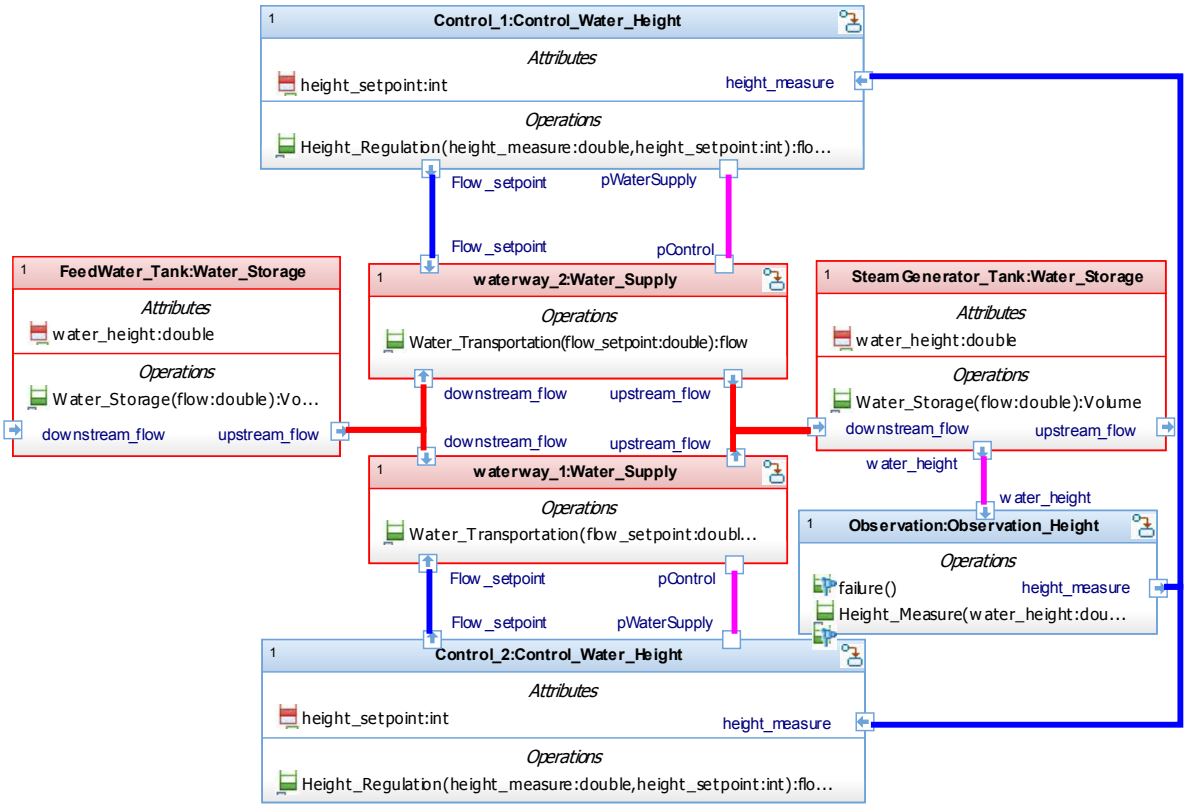

Fig. 5 Internal Block Diagram representing control functions specification (S_{AE}) (blue) of AFW process (red).

IV. CHECKING CO-SPECIFIED MODELS

Through the integrative system engineering processes, it is important to perform validation process [8]. This is in order to check that any engineering products resulting from engineering processes are compliant with given system requirements models according to the predicate (4). This is particularly true with critical systems and dependability requirements where the use of formal techniques is strongly recommended [2] by safety standards.

The dependability requirement (S_{SE3}) of our case-study is about the single failure criterion. Critical systems satisfy this criterion if their aim is ensured in the presence of any single component failure. CISPI-AFW system being used for short term during emergencies, only the single active failures must be taken into account, i.e. alignment, control and observation components. In other words, the system requirement S_{SE3} can be expressed as following: "*if a failure is present on one of components, is there an alternative (another way) to achieve the system aim?*" This led us to use reachability analysis method to check the compliance of the different specialist specification models with S_{SE3}. Indeed, the reachability analysis is a particular model checking method, which determines in a state-space (behaviour) if there is a way that respects a given property. Moreover, model checking methods allow producing a counterexample when a model fails to satisfy a desired property. "This faulty trace provides a priceless insight to understanding the real reason for the failure as well as important clues for fixing the problem" [16].

To express the property, the language that we have chosen for formal checking is Computational Tree Logic (CTL) [18] because it is sufficient to formalize the requirements of our case study into properties. The tool used is UPPAAL [19] which is an integrated tool environment for modelling and model-checking timed automata.

A. Checking of CISPI-AFW process specification (S_{PE})

For model checking and reachability analysis purposes, the dependability requirement S_{SE3} has to be refined as a formal property. During the first iteration between system specialist engineering and dependability specialty engineering, the checking should determine whether the specified CISPI-AFW process (S_{PE}) is compliant with S_{SE3}. In other words, "*if a failure is present on one of alignment*

functional components, is there a way to achieve the system aim?" This is represented by an observer (Fig.6) with an initial state, followed one state representing the occurrence of a failure and followed by another state representing the system aim satisfaction.

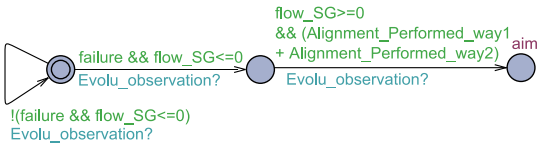

Fig. 6. Observer for process specification (S_{PE})

For each failure, the property to check is: "*Is there a way to reach the system aim after the failure occurrence?*" In the CTL, this property is written:

$$E <> (failure_i == 1) \;\&\&\; (Observ.aim) \qquad (6)$$

This property must be checked considering that each process function may have a failure. Two results are possible:

- The property is satisfied: this result allows concluding that CISPI-AFW process specification (S_{PE}) is compliant with system requirement (S_{SE3}).
- The property is not satisfied: this result allows concluding that CISPI-AFW process specification (S_{PE}) is not compliant with system requirement (S_{SE3}). However, by analysing the counterexample returned by the model checker, dependability specialist is able to identify where the problem is.

Note that the checking of this property, allows also checking the compliance with systems requirements S_{SE1} and S_{SE2}. Indeed, from a point of view of the process, we assume that if the upstream flow of the SG tank is greater or equal to the downstream flow ($20\ L.mn^{-1}$), the system can ensure to maintain the level in the SG tank (Fig.6). This is implemented in automaton diagram by the guard: *flow_SG>=0 && (Alignment_Performed_way1 + Alignment_Performed_way2)* where *flow_SG = downstream_flow_SG – upstream_flow_SG* (Fig. 9).

Thus, with SE problem space and DE solution space knowledge (K_PS_{SE}, K_SS_{DE}) we are able to check the compliance of CISPI-AFW process specification (S_{PE}) with systems requirements (S_{SE}) according to the instance of entailment (4):

$$K\_PS_{SE}, S_{PE}, K\_SS_{DE} \vdash S_{SE} \qquad (7)$$

To perform the checking, the specified behavior of functional process components must be modeled in UPPAAL. For this, the DE solution space has to translate behavior described by SE problem space in SysML statechart diagram into timed automata diagrams. Note that for reasons of size, the models presented in this paper do not represent the complete system. We focus for this study to the alignment function (Fig.7 and Fig.8) and to the formalization of the logical architecture (Fig.9). Many studies deal with the passage from statechart to automata diagrams [20] [21]. However, to apply these rules, several iterations between dependability and systems engineers have been necessary for the interpretation of the semi-formal SysML statechart diagram. Thus, each SysML states are represented by automaton states. Note that the state *Operational* does not appears in the automaton but it includes in the states {*Aligned, Misaligned, Aligning, Misaligning*}. Moreover, in the automaton diagram the state *Init_State* is included in the state *Init*. Therefore we add two transitions to the state *Init* to initialize the alignment component either in state *Aligned* or in state *Misaligned*.

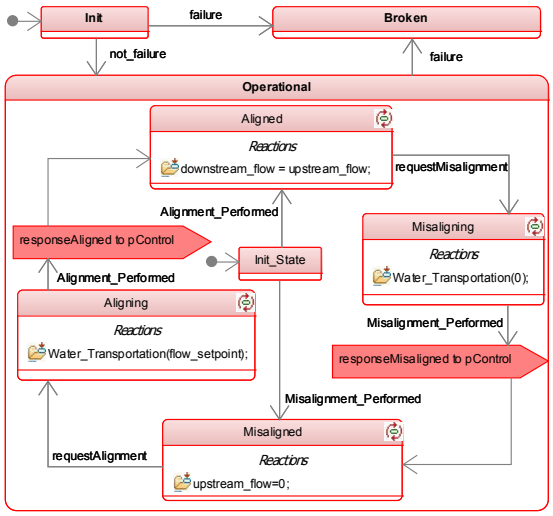

Fig. 7. SysML statechart diagram of alignment function (Water_Supply)

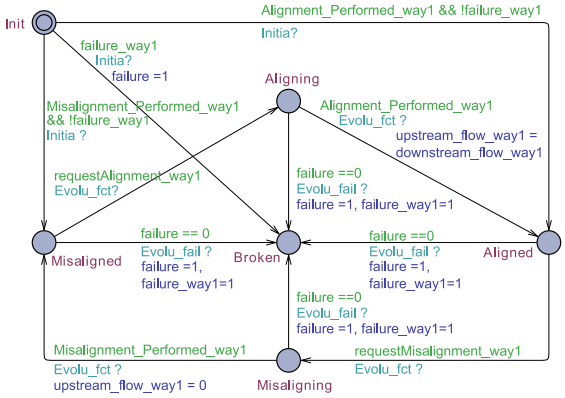

Fig. 8. Timed automata diagram of alignment function (Water_Supply)

Finally, the specified process architecture (red in Fig.5) needs to be also modeled by an automaton (Fig.9). This automaton represents the link between functional process components through water flows exchanged between them. Moreover, it allows updating water flow values at each evolution of the system.

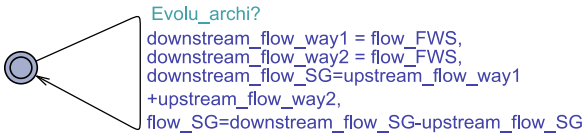

Fig. 9. Logical architecture of AFW process represented in timed automaton

From these models, we are able by model checking to check that the property (8) is satisfied. Therefore we can affirm that the CISPI-AFW process specification S_{PE} is compliant with systems requirements S_{SE} according to predicate (9).

B. *Checking of CISPI-AFW control functions specification (S_{AE})*

As for the checking of process specification (S_{PE}), the same approach can be applied to check the compliance of control function specification (S_{AE}) with systems requirements (S_{SE}) according to the instance of predicate (4):

$$K\_PS_{SE}, S_{AE}, K\_SS_{DE} \vdash S_{SE} \qquad (8)$$

The property (8) to check is the same as previously, however the observer has to be redefined to take into account control and observation functions. Indeed, the satisfaction of S_{SE1} requires to observe and to control process in order to maintain level in SG tank. Thus, the observer (Fig.10) consists on an initial state, followed of a state representing the occurrence of a failure, and followed by another state representing the system aim satisfaction if system is able to control and to observe the water level in the SG tank.

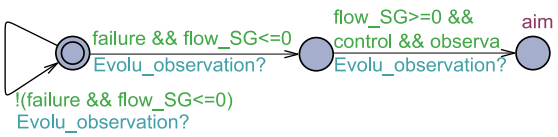

Fig. 10. Observer for control specification (S_{AE})

Following the same method as previously, dependability engineer transforms statechart diagrams of control and observation functional components into automaton diagrams. Diagrams are not presented in this paper for space limitations. Once all models have been updated, dependability engineer checks the property (8) for each components failure. For functional control component failure, the single failure criterion is satisfied. Indeed, if a failure occurs on *Control_1*component, the *Control_2* component on waterway 2 (with its associated process) components can ensure the CISPI-AFW system aim. As against, for functional observation component, if a failure occurs there are no other functional observation component to observe the water level in SG tank. Therefore, it is no longer possible to ensure the CISPI-AFW system aim. In this sense, DE solution space prescribes to systems engineering problem space that the AE specification (S_{AE}) is not compliant with systems requirements (S_{SE}). It prescribes also that another functional observation components is required to ensure the compliance of S_{AE} with S_{SE}. A new iteration between SE problem space and AE solution space is necessary to satisfy the predicate (10).

V. CONCLUSION AND PERSPECTIVES

We would point out that UPPAAL is not yet integrated into our co-simulation framework but it could be in future works. This means that the verification/validation process remains human-based (properties formalization, models transformation) with the aid of an automated checker. On contrary, it is important to emphasize the role of the co-specification process to aid also human checking modeller to ensure the correctness of its own dependability engineering models through several iterations with the system engineer in order to exhibit the right properties to the satisfaction of S_{SE}. A future work will focus on the use of SCADE® tool for checking with the advantage to automatically import IBM Rhapsody® statechart diagrams.

Others ongoing works continue to experiment the powerful role of this co-specification framework based on the quest of knowledge from the systems engineering domain to others specialist-engineering domains. For example, we are working with human-centred engineering (4) in order to satisfy operational requirements when field operators perform some control tasks but also with automation engineering in order to define a new kind of intelligent controller for CISPI-AFW [22]. A future work will also improve this co-specification process to fully define a logical architecture and then a technical one.

All these works, sometimes exploratory, are based on sound modelling rather than drawing for systems engineering, and aim in-fine to contribute to Human System Integration, both in operation and Design, as addressed by [23].

REFERENCES

[1] E. Hollnagel and D. D. Woods, Joint cognitive systems: Foundations of cognitive systems engineering: CRC Press, 2005.

[2] [G. Fanmuy, A. Fraga, and J. Llorens, "Requirements verification in the industry," in Complex Systems Design & Management, ed: Springer, 2012, pp. 145-160.

[3] F. Bouffaron, D. Gouyon, D. Dobre, and G. Morel, "Revisiting the interoperation relationships between Systems Engineering collaborative processes," presented at the INCOM 2012, 14th IFAC Symposium on Information Control Problems in Manufacturing, Bucharest, Romania, 2012.

[4] F. Bouffaron, J.-M. Dupont, F. Mayer, and G. Morel, "Integrative construct for Model-Based Human-System Integration: a case study," presented at the World Ifac Congress 2014, Cape Town - South Africa, 2014.

[5] D. Casada, "Auxiliary feedwater system aging study," Nuclear Regulatory Commission, Washington, DC (USA). Div. of Engineering; Oak Ridge National Lab., TN (USA)1990.

[6] R. Belles, J. Cletcher, D. Copinger, B. Dolan, J. Minarick, and M. Muhlheim, "Precursors to potential severe core damage accidents: 1998, A status report," NUREG/CR, vol. 4674, 1998.

[7] J. Duco, "Accidents nucléaires. Three Mile Island (Etats-Unis)," Techniques de l'ingénieur. Génie nucléaire, pp. BN3883-1, 2004.

[8] A. Pyster, D. Olwell, N. Hutchison, S. Enck, D. Anthony, H. Squires, and A. Squires, "Guide to the Systems Engineering Body of Knowledge (SEBoK). Version 1.0.1," Hoboken, NJ: The Trustees of the Stevens Institute of Technology ©2012. Available at: http://www.sebokwiki.org/index.php/article_title, 2012.

[9] A. Faisandier, collection : Engineering and Architecing multidisicplinary systems, Sinergy'Com, From 2012

[10] D. Dobre, G. Morel, and D. Gouyon, "Improving human-system digital interaction for industrial system control: some systems engineering issues," in 10th IFAC Workshop on Intelligent Manufacturing Systems, IMS'10, 2010.

[11] M. Jackson, Problem frames: analysing and structuring software development problems: Addison-Wesley, 2001.

[12] Z. Jin, "Revisiting the meaning of requirements," Journal of computer science and technology, vol. 21, pp. 32-40, 2006.

[13] OMG, OMG Systems Modeling Language (OMG SysML) (v 1.2), 2010.

[14] T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, and D. Neumerkel, "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models," in 9th International Modelica Conference, 2012.

[15] Bertsch, C., & Schulmeister, E. A. U. "The Functional Mockup Interface-seen from an industrial perspective".10 th International Modelica Conference, Lund, Sweden, 2014

[16] C. Cassandras and R. Braatz, "People In Control," IEEE Control Systems Magazine, vol. 31, p. 24, 2011.

[17] E. Clarke, O. Grumberg, and D. Peled, Model checking: The MIT Press, 1999.

[18] E. Clarke and E. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic: Springers, 1982.

[19] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in Formal methods for the design of real-time systems, ed: Springer, 2004, pp. 200-236.

[20] A. Knapp, S. Merz, and C. Rauh, "Model checking timed UML state machines and collaborations," in Formal Techniques in Real-Time and Fault-Tolerant Systems, 2002, pp. 395-414.

[21] W. Ji, D. Wei, and Q. Zhi-Chang, "Slicing hierarchical automata for model checking UML statecharts," in Formal Methods and Software Engineering, ed: Springer, 2002, pp. 435-446.

[22] M. Fliess and C. Join, "Model-free control," International Journal of Control, vol. 86, pp. 2228-2252, 2013.

[23] G. Boy and J. M. Narkevicius, "Unifying Human Centered Design and Systems Engineering for Human Systems Integration," in Complex Systems Design & Management CSD&M 2013, Paris, 2013.