# Specifying some key SE training artifacts

David Gouyon, Fabien Bouffaron, Gérard Morel

# Specifying some key SE training artifacts

**David Gouyon*, Fabien Bouffaron*, Gérard Morel***

** Nancy Research Centre for Automatic Control (CRAN), Université de Lorraine, CNRS UMR 7039*
*Campus Science, BP 70239, 54506 Vandoeuvre-lès-Nancy Cedex, France*
*{david.gouyon, fabien.bouffaron, gerard.morel}@ univ-lorraine.fr*

**Abstract:** Training Systems Engineering (SE) is increasingly spreading in academic curriculums in order to satisfy the growing need of engineers aware of systems view. This is achieved by the formalization of ad-hoc best practices into a more mature corpus expecting to make SE a full discipline. However, one major training difficulty is to infuse multi-disciplinary views of a system as a whole beyond implementing standardized engineering processes. In this way, our training practice leads us to well formalize the specification process as a basic driver in order to logically guide both trainers and trainees SE practices.

## 1 Introduction

There is a growing interest in Systems Engineering (SE) [Haskins et al. 2011], as the discipline of engineering a system as a whole [Von Bertalanffy, 1968]. The objective is to address the increasing complexity caused technically by multiple system component architectures as well as caused behaviorally by multiple interacting agents under control. Classical engineering disciplines have to broaden their respective theoretical and technical domains to meet this crosscutting challenge in order to define, to develop and to deploy a system satisfying stakeholder requirements.

For SE to be recognized as a full discipline by the industrial and academic worlds, the BKCASE[1] international project (Body of Knowledge and Curriculum to Advance Systems Engineering) aims at proposing a worldwide knowledge repository, composed of two major deliverables. The first one is the SEBoK (Systems Engineering Body of Knowledge) [Pyster et al. 2012a], which purpose is to "provide a widely accepted, community based, and regularly updated baseline of SE knowledge". It is available as a wiki which describes precepts about Systems Science, System Thinking, and mainly about Systems Engineering. The second deliverable is the GRCSE (Graduate Reference Curriculum for Systems Engineering) [Pyster et al. 2012b], which is a set of "recommendations for the development and the implementation of a systems-centric professional master's degree program in Systems Engineering", compliant with the SEBoK.

With the same objective than the SEBoK, the AFIS[2], the French chapter of the INCOSE[3], has proposed in parallel a SE reference book [Fiorèse & Ménadier 2012]. The presentation approach used is to describe basic concepts ("with what?", "why?", "what?"), before describing SE processes, methods and tools ("how?"). Our eight years of experience in teaching SE within a master's degree in Complex Systems Engineering[4] convinced us that, indeed, it is important for students to understand first how SE basic precepts are structured. In this sense, considering the SE domain as the pivotal domain between the trainer domain and concurrent specialist engineering domains (Fig. 1) [Bouffaron et al. 2012], we prescribe that satisfying systems requirements as well as functionally and physically architecting a solution rely mainly on a **logical ORM-based[5] specification process**. The objective of this paper is to propose SE training recommendations based on this specification process as driver, before to assess them on Model-Based Systems Engineering (MBSE), in particular with SysML [OMG 2012].

David Gouyon, Fabien Bouffaron, Gérard Morel
Centre de Recherche en Automatique de Nancy
Université de Lorraine, CNRS
Boulevard des aiguillettes, BP 70239, 54506 Vandoeuvre les Nancy, France
e-mail :{david.gouyon,fabien.bouffaron,gerard.morel}@univ-lorraine.fr

[1] http://www.bkcase.org

[2] http://www.afis.fr

[3] http://www.incose.org

[4] http://formations.univ-lorraine.fr

[5] http://www.ormfoundation.org/. Note that ORM enables to verbalize our specified training facts

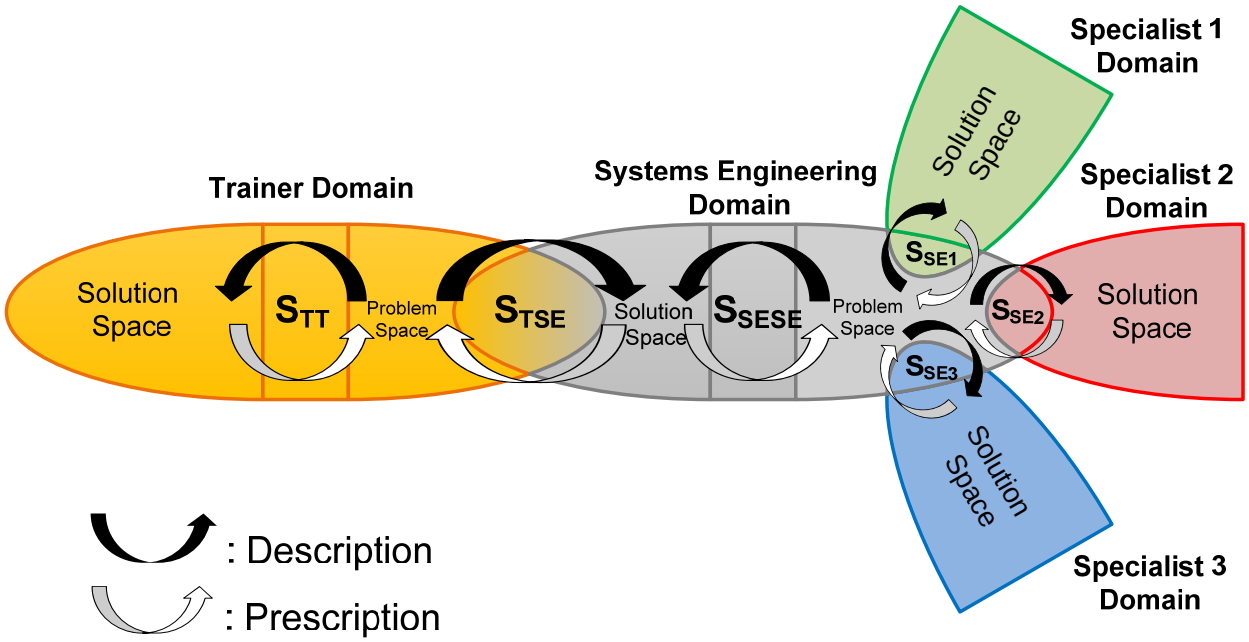


**Fig. 1 Collaborative specification process applied to SE training**

## 2 SE training problem statements

Systems Engineering teaching is gradually being implemented in some schools and universities, but is not unified. It is often taught by experienced engineers and professors, who are rather autodidacts and trained by years of practices on large projects. These engineering projects pushed them to cope with different types of problems, to find solutions, and to formalize "best practices" to implement on future projects. As a result, SE approaches are mostly based on feedbacks rather than on sound scientific foundations. Furthermore, in function of trainers' experience and tendency, some complementary points of views can be followed, with advantages and drawbacks, to teach SE: ensuring compliance with SE standards, following MBSE methods, and applying project management concepts.

Indeed, a major effort has been done since years to standardize SE processes (ANSI/EIA 632 [ANSI/EIA 1999], IEEE 1220 [IEEE 2005], ISO/IEC 15288 [ISO/IEC 2008], ISO/IEC TR 24748-2 [ISO/IEC 2011], ISO/IEC 26702 [ISO/IEC 2007]). As a result, agreement, enterprise, project and technical processes involved within system life cycle are highlighted and defined. This enables to realize which processes have to be applied, how they can be applied, and what their outcomes are. This way, trainees can verify if they have well applied SE processes in order to "solve the problem right", but the main drawback is that the compliance to these processes does not guarantee that they have worked "the right problem" [Martin 1997].

The implementation complexity of these standardized processes grows in function of the number of systems functionalities, which increases rapidly with the growing use of software. The traditional document-centric approaches come then to theirs limit [Pyster et al. 2012 a]. One answer seems to be Model-Based Systems Engineering (MBSE) [Estefan 2008], which seems to be better suited to manage complexity, including that of software [Friedenthal et al. 2011]. The objective of MBSE is to support system life cycle activities, including requirements engineering, high-level architecture, detailed design, testing, usage, maintenance, and disposal, with models. In practice, if trainees are not well guided to apply MBSE, graphical representations are used to replace traditional textual documents. The result is a set of diagrams representing aspects of the system, but which can present inconsistencies. In this case, students have only "drawn" rather than "modeled".

Even if the management of SE processes can not be dissociated from project management [PMI 2008], it is also clear that a senior project manager (Fig. 2a or Fig. 2c) has previously managed technical processes in several projects as junior manager (Fig. 2b). In other words, the trainer is in charge of the strategic project management according to a training development cycle, while trainees are in charge of the operational management of SE tasks as well as of their execution.

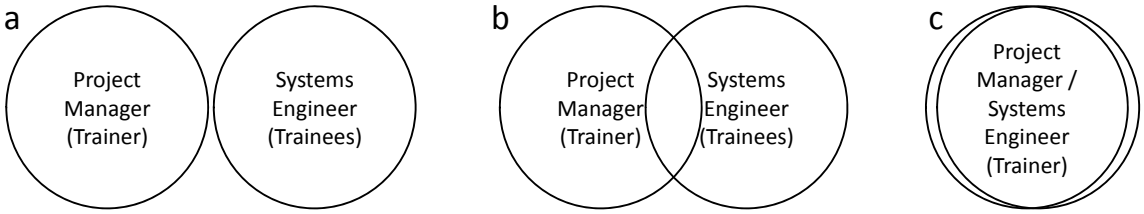


**Fig. 2 Overlap of the project roles, adapted from [Pyster et al. 2012]: trainer and trainees share the project processes**

This interoperation relationship between trainer and trainees (Fig. 3) must be also contextualized in a common domain-of-interest, from which the system-of-interest must be specified. This system-of-interest must have a scale factor and a complexity justifying that various disciplines (system, specialties…) are involved as well as that different roles are allocated to trainees.

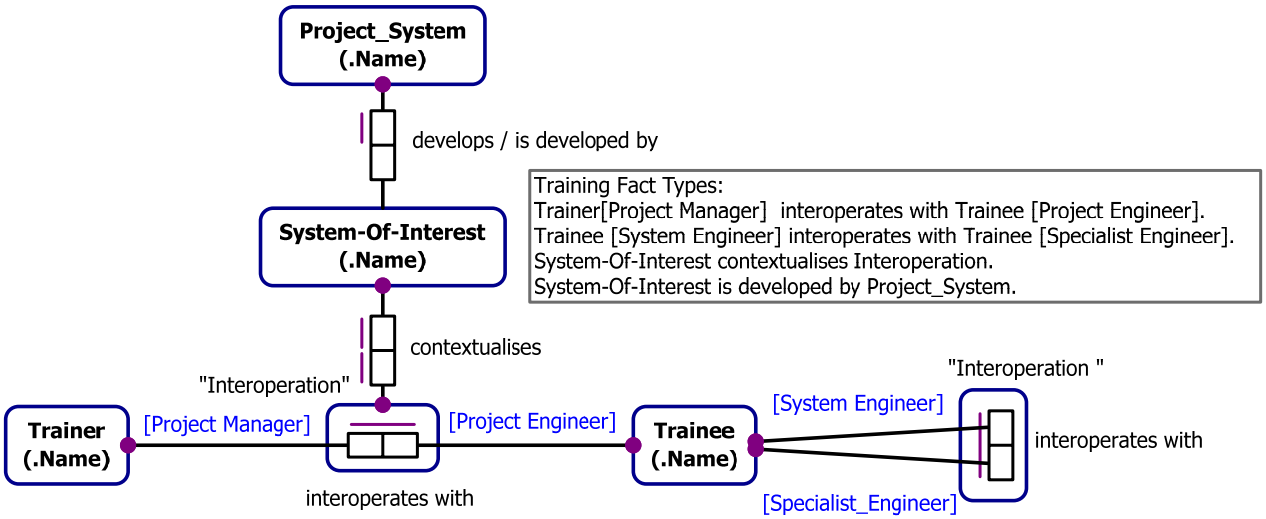


**Fig. 3 Interoperations between trainers and trainees within SE training projects**

As example of training context, the CISPI lab platform reflects some requirements of the large-scale R&D programme aiming to innovate in power-plant control system-architectures for technical as well as ecological issues. CISPI system-of-interest reflects the main principles of a steam generator Auxiliary Feedwater System (AFS). The purpose of such an AFS is to participate in the cooling of the primary circuit, in case of emergency. Its mission is to feed steam generators with water in order to get the necessary conditions for the use of a Shutdown Cooling Heat Exchanger (SCHE). The main objective is to maintain a sufficient water flow rate into the steam generators, so that they can ensure heat transfers in safe conditions. Student training projects impact modifications of the current CISPI platform related to an integrated Control, Maintenance and technical Management System [Morel et al. 2009] as well as fluid circuits.

## 3 Specification as a SE training driver

The problem statements we pointed out in the previous section brought us to formalize the contractual specification relationship between trainers and trainees, as a pivotal driver to logically organize SE artifacts within training projects.

## *Problem-Solution spaces interoperation*

In Fig. 3, Trainer$_{[Project_Manager]}$ and Trainee$_{[Systems_Engineer]}$ can be associated to *problem spaces*. Trainee$_{[Project_Engineer]}$ and Trainee$_{[Specialist_Engineer]}$ can in turn be associates to *solution spaces*. Indeed, problem space *describes* its *problem* and solution space *prescribes* a *solution* in return. If the solution proposed solves the problem, this solution is seen as a *specification result.* In this sense, as underlined by [Hall et al., 2002] the interoperation between problem space and solution space is seen as a *specification process* producing this specification result. This is consistent with previous work [Bouffaron et al., 2012] wherein the specification process is seen as a descriptive-prescriptive interoperation relationship between problem space and solution space. This formalization (Fig. 4) allows clarifying the use of the "specification" word (which is used for referring to the *specification process* as well as the result of this process [Van Lamsweerde, 2000]) and the emergence of a key SE artifact which is the separation of problem and solution spaces.

Thus, **Trainee$_{[Systems_Engineer]}$ and Trainee$_{[Specialist_Engineer]}$ would not be the same person**. Therefore, **trainees would be specialized in either system engineering domain, or in specialist domains.**

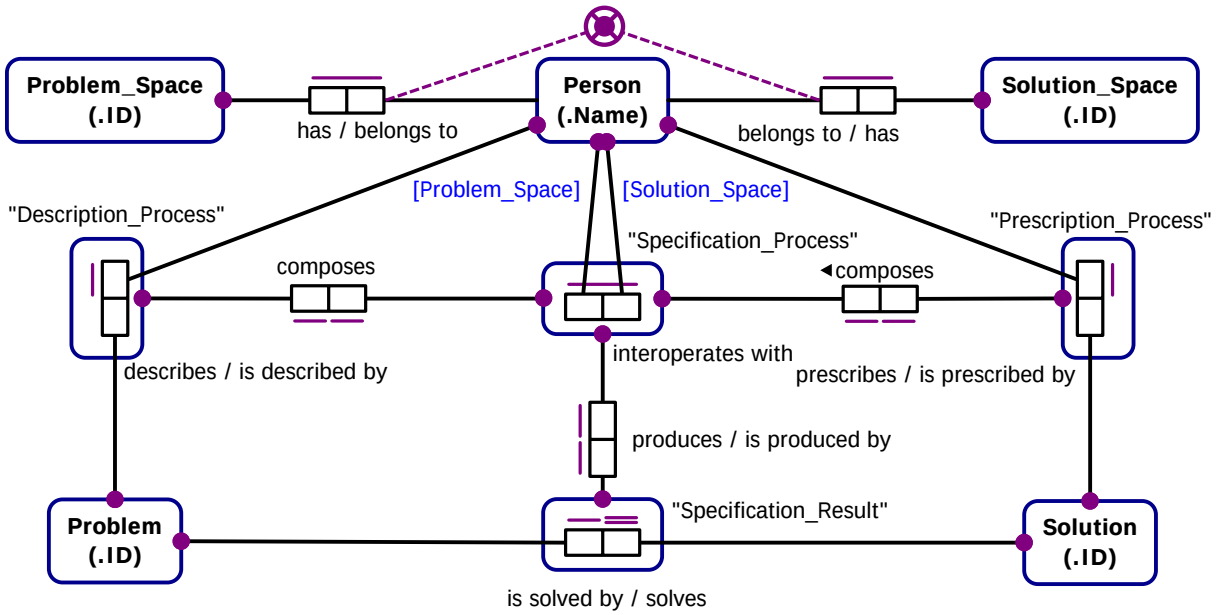


**Fig. 4 Problem-solution spaces interoperation artifacts**

## *Source-Sink objects interoperation*

In a more general way, the interaction between two spaces highlights that the specification process treats stakeholder requirements (Problem) as *source* objects, to specify system requirements (Solution) as *sink* objects. We noticed that the specification process evolution is clocked by the different roles of the objects handled by the processes composing the specification process: an object produced by a process with a sink role will have a source role when it will be consumed by another process (Fig. 5). A main interest in the Source-Sink artifact is that it can be useful to perform traceability between source objects and sink objects during process execution. In this sense, **trainer and trainee would trace all actions performed during process specification execution to ensure traceability between problem and solution**.

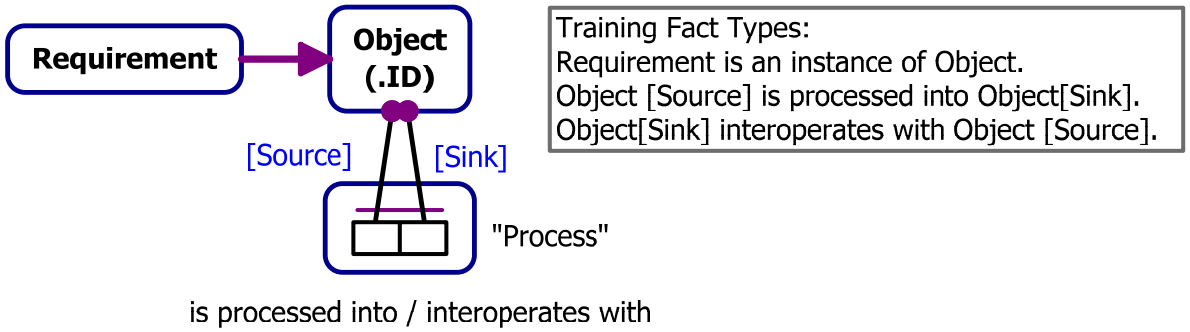


**Fig. 5 Source-sink objects interoperation artifacts**

### *Optative and indicative moods interoperation*

The process formalization (Fig. 6) can be put in relation with works by Jackson in the software engineering domain, who formalized the concepts of optative and indicative moods [Jackson, 1997]. During the specification process, *optative objects* (representing requirements at different levels of abstraction) undergoes several transformations performed by *processes* according to *indicative objects* belonging either to the problem space or to several specialist solution spaces. *Optative object* expresses conditions over the problem space that have to become true. *Indicative objects* represent the known properties (skills) of a domain which are validated by experts regardless of the behavior or given properties of the solution. Thus, domains can act as solution spaces if they have skills to solve a problem, or as problem space if they don't [Czarnecki, 1998]. This highlights that **some students would be involved at the system level, and some other students would be specialized in engineering domains**.

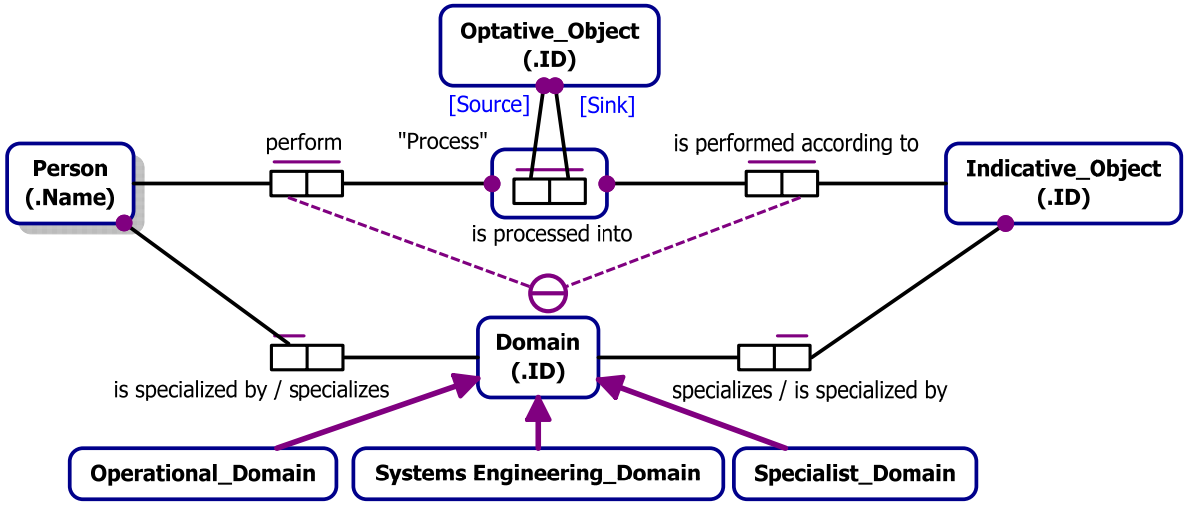


**Fig. 6 Optative and indicative moods interoperation artifacts**

### *Verification and Validation processes interoperation*

The relationship between problem and solution spaces appears as a contractual process involving validation and verification [Pyster et al., 2012a] (Fig. 7). "Validation is used to ensure that one is working the right problem, whereas verification is used to ensure that one has solved the problem right" [Martin, 1997]. In this sense, we propose a formalization of the verification and validation processes by interpreting Jackson's works [Gunter et al., 2000] about optative and indicative moods, and con-

sidering the predicate: $W \wedge S \Rightarrow R$, where the specification S (Optative object Sink) must satisfy the requirement R (Optative object Source) considering the domain knowledge W (Indicative Object). Thus, the verification process performed by solution space consists in the satisfaction of the predicate:

$$W_{solution\ space} \wedge Optative\_object_{Sink} \Rightarrow Optative\_object_{Source}$$

In a similar way, the validation process performed by problem space has to satisfy the predicate:

$$W_{problem\ space} \wedge Optative\_object_{Sink} \Rightarrow Optative\_object_{Source}$$

This shows that **trainees would not valid by themselves their solution**. **Their work would be evaluated by the corresponding problem space**. It can be either the trainer for the system level, or other trainees for specialists.

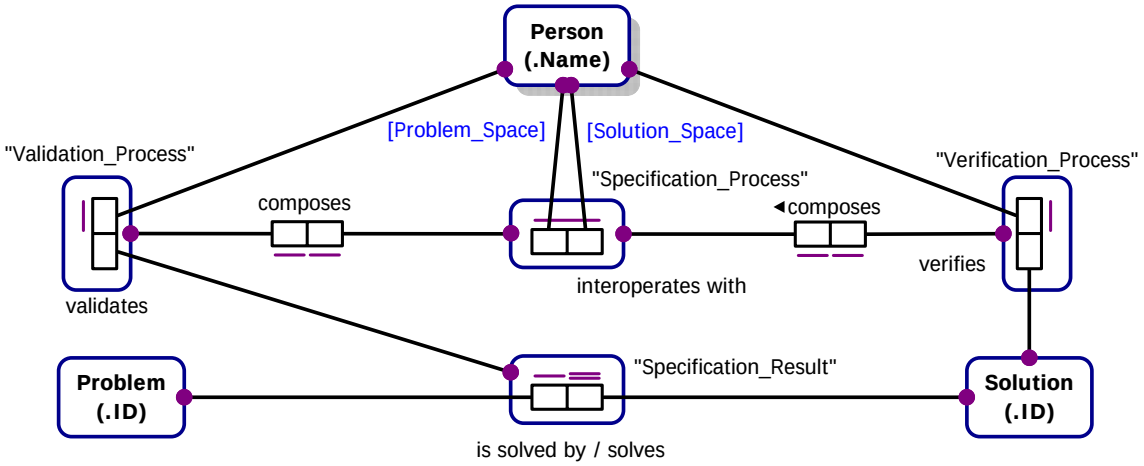


**Fig. 7 Verification and Validation processes interoperation artifacts**

## 4 SE training solution assessments

To illustrate the potential benefits of the recommendations expressed in the previous section, we focus on the specification of the CISPI platform modifications. In order to specify it, a set of projects are proposed by relevant trainers (problem space) to student teams (solution spaces), as parts of their SE training curriculum.

The first set of observations we make is that, with such formalizations and explanations, trainees understand more easily SE concepts and the relative positioning of SE processes. They are aware of their respective roles within problem and solution spaces, and establish contractual interoperations they should have between them, and with trainers. They clearly make the difference between requirement definition and analysis processes, and between verification and validation processes.

The second set of observations we make concerns the application of the recommendations on MBSE, more specifically with SysML [Holt & Perry 2008]. We proposed to students to interpret SE recommendations into SysML modeling recommendations or rules, in order to facilitate diagram authoring and to improve SysML semantics [Ober et al. 2011]:

- The problem / solution spaces partitioning of SysML models can be done using packages: “problem space package” and “solution space package”. This has a main interest for requirement definition and analysis, for example to clearly separate stakeholder requirements and system requirements;
- The source & sink concept is closed to UML customer & supplier roles used in dependencies to ensure traceability between objects. Given that SysML is a UML profile, we propose to use dependencies for the traceability during the requirement specification process, as presented in Table 1. Note that during the requirement analysis process, we have identified 4 types of transformations: refinement, decomposition, composition and induction [Bouffaron et al., 2012];
- As transformations rely on skills, it is very important to trace their use in models. We propose to include skills into models using SysML “Rationales”. Such rationales can be linked to dependencies between requirements to justify and trace the transformation performed.
- Although verification and validation processes can be executed according to skills, which can be traced using SysML rationales as presented before, they are usually performed according to SysML “Test Cases”. Problem and solution spaces do

not have necessarily similar test cases to verify and validate requirements. Thus we propose to link solution space test cases to requirements with a “verify” dependency as the indicative mood for the verification process, and to create the “validate” dependency for the validation process in order to clearly make the difference between verification and validation.

| Process | Source | Sink | SysML dependencies |
|---|---|---|---|
| **Description** | Stakeholder requirement – Problem space | Stakeholder requirement – Solution space | trace |
| **Prescription** | System requirement – Solution space | System requirement – Problem space | trace |
| **Requirement Analysis** | Requirement level n | Requirement level n+1 | Refine, Derive, Requirement containment relationship … |
| **Validation** | Stakeholder requirement | System requirement | satisfy |

**Table 1. SysML dependencies used to model source and sink relations**

To illustrate this second set of observations, we focus on the following stakeholder requirement, extracted from a student project: “CISPI-AFS shall produce a sufficient water flow rate for the return of the primary circuit to SCHE conditions”. Considering the work of students who are not guided by the recommendations made in section 3, the resulting requirement diagram is very poor, presenting only a stakeholder requirement and a system requirement tracing it (Fig. 8).

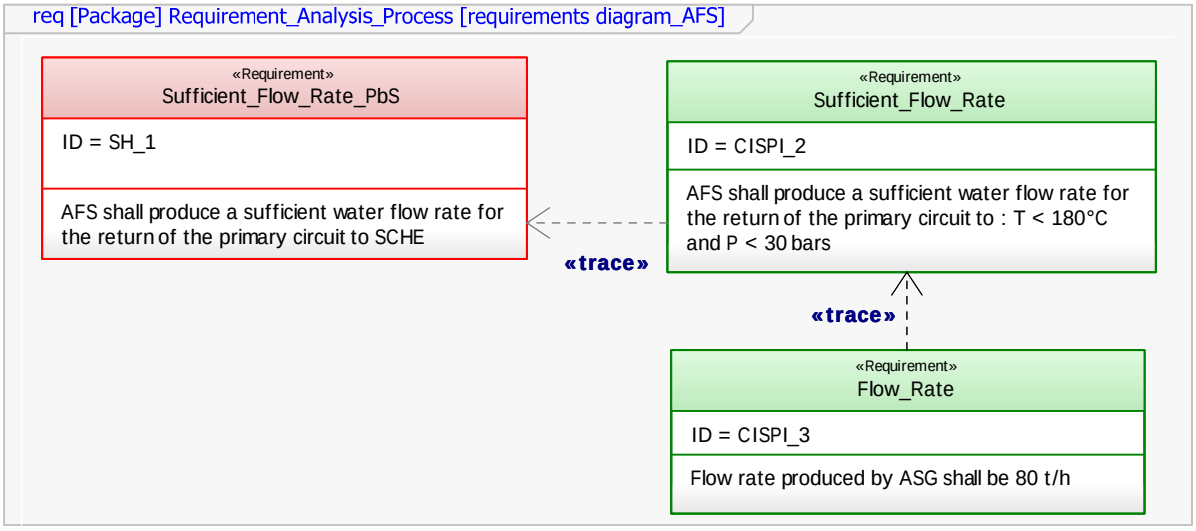


**Fig. 8 Requirement diagram made by students without rules application**

Considering now the work done by students guided by section 3 recommendations (Fig. 9), the approach is the following: once expressed in the *trainer problem space* (represented by a package), the stakeholder requirement is described to the *trainees' system level solution space* (also represented by a package). To transform (refine) this requirement into system requirements, as well as to verify the requirements they produce, trainees need skills that they can have or that can be required to specialist engineering solution spaces. These skills are then traced in models with rationales. The validation of system requirements is done by the trainer in the problem space using a test case which is traced in models.

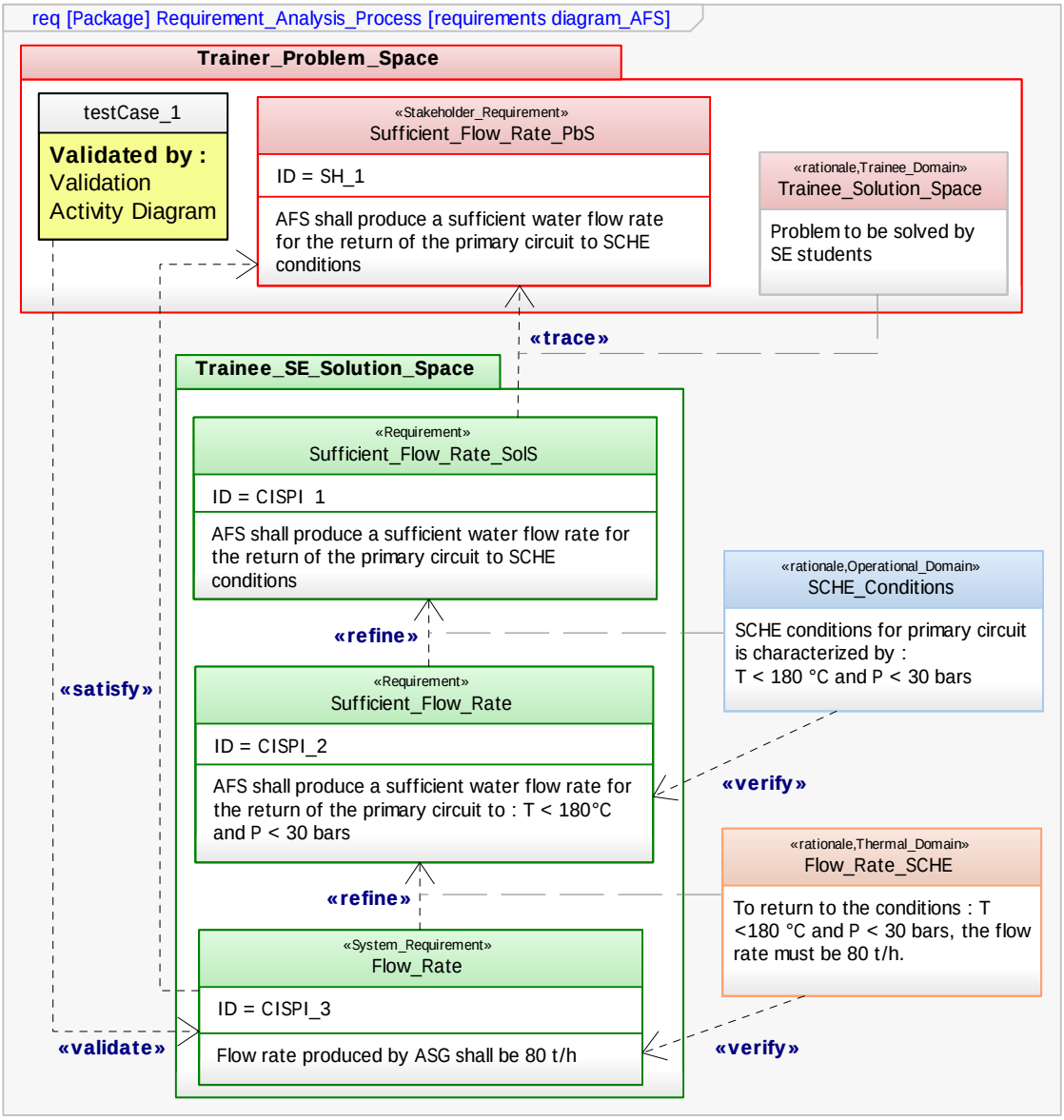


**Fig. 9 Requirement diagram made by students applying rules**

The comparison of the requirement diagrams of Fig. 8 and Fig. 9 shows that recommendations improve the traceability because requirements are clearly classified into packages, and because skills used for requirement transformations and verification are traced into models. The verification of student work is also simpler, which is important in a learning context in which teachers can be at the same time the contracting authority and the support of student assessment. More detailed dependencies links (verify validate…) helps the mutual understanding of models, and accelerates the specification process in decreasing the number of iterations before converging on validated system requirements.

Note that this training process based on a reference specification process enable to rationally extend the project structure of SysML based tools as the one[7] for the CISPI project, as well as their metamodel for the relevant use of required SE artifacts.

[7] IBM® Rational® Rhapsody® supporting Model transformation Based Systems Engineering processes.

## 5 Conclusions

In order to cope with the large amount of “best-practices” required to face any SE project with the limited amount of training resources (time, platforms, trainers’ skill and knowledge, trainees’ knowledge heterogeneity…), we decided to share a common model of understanding of the process of engineering a system as a whole. This is a prerequisite to logically train mainly the technical processes and their related SE artifacts based on the specification process considered as a key SE driver.

ORM diagrams presented in this article are parts of a metamodel under development for training as well as for engineering purposes. This SE specification-based metamodel will be used as a pivotal reference in order to map systems modeling languages and tool artifacts with best SE artifacts.

## 6 References


ANSI/EIA (1999). ANSI/EIA-632. Processes for engineering a system. Electronic Industries Alliance, Government Electronics And Information Technology Association Engineering Department, EIA Standard.

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

Czarnecki K. (1998). Generative programming: Principles and techniques of software engineering based on automated configuration and fragment-based component models. PhD thesis, Technical University Of Ilmenau, Germany.

IEEE (2005). 1220. IEEE standard for application and management of the systems engineering process. IEEE Computer Society.

Estefan J. A. (2008). Survey of model-based systems engineering (MBSE) methodologies. Incose MBSE Focus Group, 25.

Fiorèse S. and Meinadier J.-P. (2012) To discover and understand Systems Engineering (in French). ISBN: 978.2.36493.005.6. Cépaduès,

Friedenthal S., Moore A., and Steiner R. (2011). A practical guide to SysML: the systems modeling language. ISBN: 978-0123852069. Morgan Kaufmann

Gunter C.A., Gunter E.L., Jackson M., and Zave P. (2000). A reference model for requirements and specifications. IEEE Software, 17, pp. 37-43.

Hall J.G., Jackson M., Laney R.C., Nuseibeh B., and Rapanotti L. (2002). Relating software requirements and architectures using problem frames. In IEEE Joint International Conference on Requirements Engineering. Essen, Germany.

Haskin C., Forsberg K., Krueger M., Walden D., Hamelin R. D. (eds.) (2011). Systems Engineering Handbook, A guide for systems life cycle processes and activities, V. 3.2.2. INCOSE.

Holt J. and Perry S. (2008). SysML for Systems Engineering Vol. 7. Inst of Engineering & Technology.

ISO/IEC (2007). ISO/IEC 26702. Systems engineering - Application and management of the systems engineering process.

ISO/IEC (2008). ISO/IEC 15288. Systems and software engineering - System life cycle processes.

ISO/IEC (2011). ISO/IEC TR 24748-2. Systems and software engineering - Life cycle management - Part 2: Guide to the application of ISO/IEC 15288 (System life cycle processes)

Jackson M. (1997). The meaning of requirements. Annals of Software Engineering 3, pp. 5-21.

Martin J. N. (1997). Systems Engineering Guidebook: A process for developing systems and products. ISBN: 978-0849378379. CRC Press.

Morel G., Pétin J.-F. and Jackson T. L. (2009). Reliability, Maintainability, Safety, In Springer Handbook of automation, pp. 735-747, ISBN: 978-3540788300, Springer.

Ober, I., Ober, I., Dragomir, I., and Aboussodor E. A. (2011). UML/SysML semantic tunings. Innovations in Systems and Software Engineering, 7(4), 257-264.

OMG (2012). OMG Systems Modeling Language (OMG SysML) (v 1.3).

PMI (2008). A guide to the Project Management Body of Knowledge (PMBoK guide), 4th edition, ISBN: 978-1933890517. Project Management Institute.

Pyster A, Olwell D., Hutchison N., Enck S., Anthony D., Squires A. (eds) (2012). Guide to the Systems Engineering Body of Knowledge (SEBoK). Version 1.0.1. Hoboken, NJ: The Trustees of the Stevens Institute of Technology.

Pyster, A., Olwell D.H., Ferris T.L.J., Hutchison N., Enck S., Anthony J., Henry D., and Squires A. (eds.) (2012). Graduate Reference Curriculum for Systems Engineering (GRCSE™). Hoboken, NJ, USA: Trustees of the Stevens Institute of Technology.

Van Lamsweerde, A. V. (2000). Formal specification: a roadmap. Proceedings of the ACM Conference on the Future of Software Engineering, pp. 147-159.

Von Bertalanffy L. (1968). General System Theory: Foundations, development, applications. Revised edition, George Braziller.