An Interface Pattern Model for Supporting Design of Natively Interoperable Systems

Vincent Chapurlat(✉), Nicolas Daclin, and Stéphane Billaud

LGI2P, Parc Scientifique G. Besse – Ecole Des Mines D'Alès,
30035 Nîmes Cedex, France
{vincent.chapurlat,nicolas.daclin,stephane.billaud}
@mines-ales.fr

Abstract. This article focuses on the interoperability feature seen as a specific requirement. Indeed, any complex system (e.g. a train, an organisation or an IT system) need to interact with other systems, thereby forming a heterogeneous environment. All these systems are not necessarily designed to function properly and efficiently with one another, whether from a conceptual, technical, behavioural or organizational standpoint. This paper highlights what seems to be relevant in terms of conceptual definitions and modelling framework whenever a (group) of engineer(s) intends to design what we call here a "natively interoperable system" or, at least, a system maximizing its interoperability capabilities. To proceed, as a first prerequisite, a definition of the concept of interoperability is here proposed for complex system engineering. The second prerequisite consists of establishing the needs of a design team assigned to design such "natively interoperable system". An interface pattern model with sufficient generic, formal and pragmatic qualities is then proposed and illustrated briefly.

Keywords: System interoperability · Natively interoperable systems · Design for interoperability · Interface pattern model

1 Introduction

Many examples from industry have highlight that a lack of interoperability of systems leads to delays, failures, dysfunctions or shortcomings all along these systems' life cycle; problems that can be much more manageable if they are characterized and detected earlier in the system's design stage. So, various research and development were focused on interoperability management problematic particularly over the last decade considering interoperability as an essential feature of any kind of technical or socio technical complex system (e.g. a transportation system or a Collaborative Network of Organisations [1]). The goal is to design a system able to assume its mission and for this able to maximize and maintain its abilities to interact efficiently with other systems (technical or sociotechnical, more or less complex themselves) in various situations, even more or less unpredictable, throughout its life cycle and without unwanted effects that can affect the behaviour of each systems involved in the interaction. In this sense an interaction, requested or not, consists to exchange and share items from different nature (digital i.e. data/information/knowledge, physical i.e. any kind of energy field, or material e.g. raw material, product, part, or waste).

M. van Sinderen and V. Chapurlat (Eds.): IWEI 2015, LNBIP 213, pp. 171–185, 2015.
DOI: 10.1007/978-3-662-47157-9_15

Further, an interoperable system must perform efficiently its mission independently from other systems with which it must interact for achieving this mission. However, lot of systems are currently more or less designed in order to be integrated into a given and fully identified upper-level system. These practices limit drastically the analysis of the real expected system interoperability, for instance, in avoiding unexpected or reverse effects that the interface is unable to prevent or, failing this, to protect the system itself. Last, even if important recommendations and standards are now available for instance concerning Health Care systems [2], IT systems [3], or Defence systems [4] design, the notions of interoperability requirements, interoperability analysis issues, interface or interaction still remain poorly formalised in engineering activities.

This article aims to propose conceptual elements for supporting complex system design stage taking into account requested system's interoperability. The goal is to help engineers' teams to design a so-called "*natively interoperable system*". First, definitions of system interoperability and interoperability requirements, obviously, of *natively interoperable complex system* are proposed. Second, an interface pattern model is needed to face design issues of such systems. This is done adopting a set of pre-requisites conceptual and then generic definitions (processor, interaction, effect…) that can be applied to various nature of systems. The goal is to provide engineers with modelling language and, by evidence, a verification tooled approach allowing them to confirm and to check whether the considered system can avoid defects due to its interaction with other systems under all specified conditions via its interfaces.

2 System Interoperability/Interoperability Requirements

Definition of interoperability depends of the application domain[1] and authors e.g. [5–9]. The goal here is to propose and adopt (in this paper at least) a definition of system interoperability for the purpose of system design stage. Classically, interoperability is "*connecting people, data and diverse systems. The term can be defined either technically or comprehensively, in taking into account social, political and organizational factors*". Then, "*two or more devices are said interoperable if, under a set of conditions, the devices are able to successfully establish, sustain and, if necessary, break a link while maintaining a certain level of performance*". In technical systems, interoperability is "*a property of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, without any restricted access or implementation*". In socio-technical systems, it is defined as "*a property referring to the ability of diverse systems and organizations to work together*", i.e., to inter-operate.

[1] The reader can find various interoperability definitions used in this section in glossaries available on [last visited and checked 2011-04-12]: http://dli.grainger.uiuc.edu/glossary.htm, http://www.eu-share.org/glossary.html, www.csa.com/discoveryguides/scholarship/gloss.php, www.naccho.org/topics/infrastructure/informatics/glossary.cfm, ec.europa.eu/transport/inland/glossary_en.htm,www.nato.int/docu/logi-en/1997/defini.htm, www.cs.cornell.edu/wya/DigLib/MS1999/Glossary.html, www.ibtta.org/Information/content.cfm, dli.grainger.uiuc.edu/glossary.htm, cloud-standards.org/wiki/index.php, en.wikipedia.org/wiki/Interoperability, wordnetweb.princeton.edu/perl/webwn, www.anzlic.org.au/glossary_terms.html

For instance, enterprise interoperability is defined as "*a cooperative arrangement established between public and/or commercial entities (authorities, parking facility operators,* etc.*), wherein tags issued by one entity will be accepted at facilities belonging to all other entities without degradation in service performance*". In the same manner, interoperability is considered as "*the ability of a system or product to work with other systems or products without special effort from the customer*". In the military field, NATO defines also interoperability as "*the ability of systems, units or forces to provide services to - and accept services from - other systems, units or forces and to use these services so exchanged to enable them to operate efficiently together*". Last, in transportation systems, interoperability seems to be achieved when "*a transport network [is suitable] for movements without breaking bulk*".

Thus we propose defining system interoperability as:

"The set of abilities and associated capabilities of a system (namely "S" from now) that allow S to be and to stay able to exchange and work harmoniously with other systems from its upper level all along its life cycle:"

- *To fulfil a common mission (i.e. the main function for which the overall system is designed), possibly time-bounded, while remaining able to perform its own mission and to reach its own objectives through the use of exchanged items with other systems then when S is interacting with these systems whatever may be their nature;*
- *In all specified operational situations (e.g. nominal functioning mode, or functioning modes when facing a risky situation) met throughout its life cycle;*
- *Reflecting the stakeholders' requirements under every specified situation.*

This capability indicates and allows assessing before, during or after the interaction - and when placed in its environment – that S does not require or result in major changes to its operations, structure or behaviour; consequently, its functional and non-functional requirements (performance, security, safety, ergonomics, human factors, etc.*) are not altered. Moreover, this does not induce undue adverse effects (dysfunctions, risks) when S is achieving its mission independently of every other system*".

As a consequence a natively interoperable system S is "*a system designed to maximize its ability to interoperate all along its life-cycle*". During S design stage, engineer has to consider carefully this expectation and shall *"[…] ensure the compatibility, interoperability and integration of all functional and physical interfaces and then ensure that system definition and design reflect the requirements for all system elements: hardware, software, facilities, people, and data*" [10]. To this purpose, one or several interfaces are requested. An **interface** is defined in [11] as "*a boundary across which two independent systems meet and act on or communicate with each other*". That requires, at least by adhering to published interface standards or by making use of a 'broker' of services able to assume interface role between *S* and other systems, possibly, "*on the fly*". Communication, synchronization or even exchange protocols must be defined and applied.

3 Interface Elements: Prolegomena

Let's recall briefly a set of concepts from the literature on system sciences [12–14] and some theoretical foundations of Systems Engineering defined as "*an interdisciplinary approach and means to enable the realization of successful systems* [socio-technical or technical systems]. *It focuses on defining customer needs and required functionalities early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem*" [15–17].

A **processor** aims to transform **items** (*digital* i.e. data/information/knowledge, *physical* i.e. any kind of energy field, or *material* e.g. raw material, product, part, or waste) transported by input **flows**, into new items transported by output flows, under the control of other flows and by using **resources** that support or are involved in processor functioning. As an example, S is a processor, a function, an activity or a process when considering its functional view; moreover, a component or an organizational unit involved in S is a processor when considering physical view (organic, organizational).

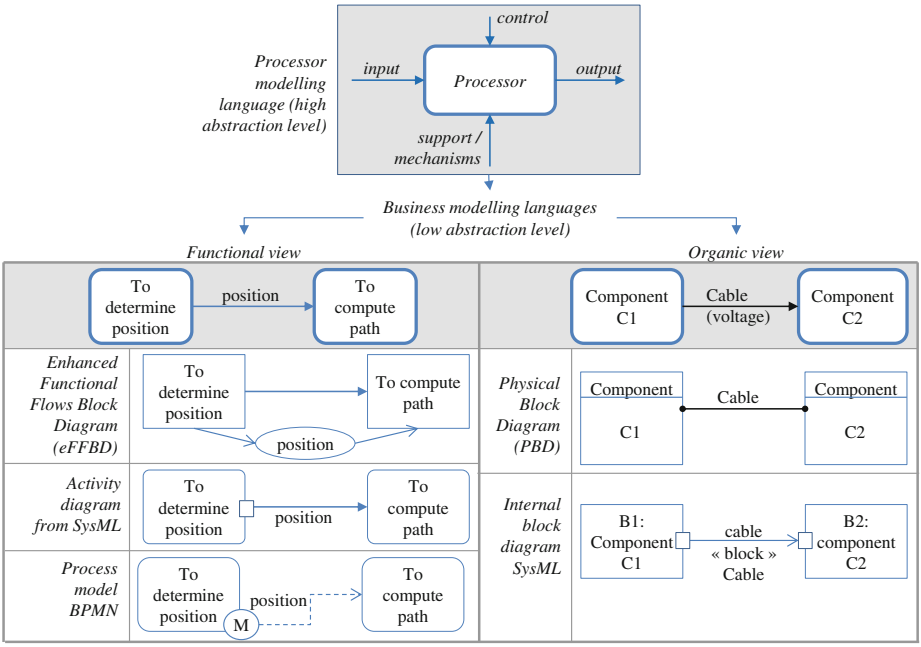

Fig. 1. From high abstraction level to low abstraction levels of modelling

Figure 1 shows the links between an abstract modelling language proposed in the SAGACE approach [13] and some equivalent notations used in various domains such as eFFBD (enhanced Functional Flows Block Diagram) [18], PBD (Physical Block Diagram), BPMN 2.0 (Business Processes Modelling Notation) [19], Activity or Internal Block Diagram from SysML [20]. So, all of the proposed concepts discussed below can be applied independently of the adopted modelling language. This step

offers a freedom to the designer in choosing the most relevant modelling language when addressing system S interfaces design.

The processor behaviour is described by a **transformation** that details the inputs flows/outputs flows treatment provided by the processor. This transformation may be described by modelling the modification induced by the processor on one or more characteristics of each item transported by input flows, so as to obtain new or modified items transported by output flows. More generally, the characteristics of any concept are named formally Space, Shape, Time attributes i.e. **SST attributes** in the next: Space (e.g. type, definition domain, instantaneous value/default value…), Shape (e.g. optical, electromagnetic, signal, binary, or linked to the aspect of the pointed out item if it can be considered as dependent from one or several of five senses) and Time (update frequency, maximum life cycle before updating…) (Fig. 2).

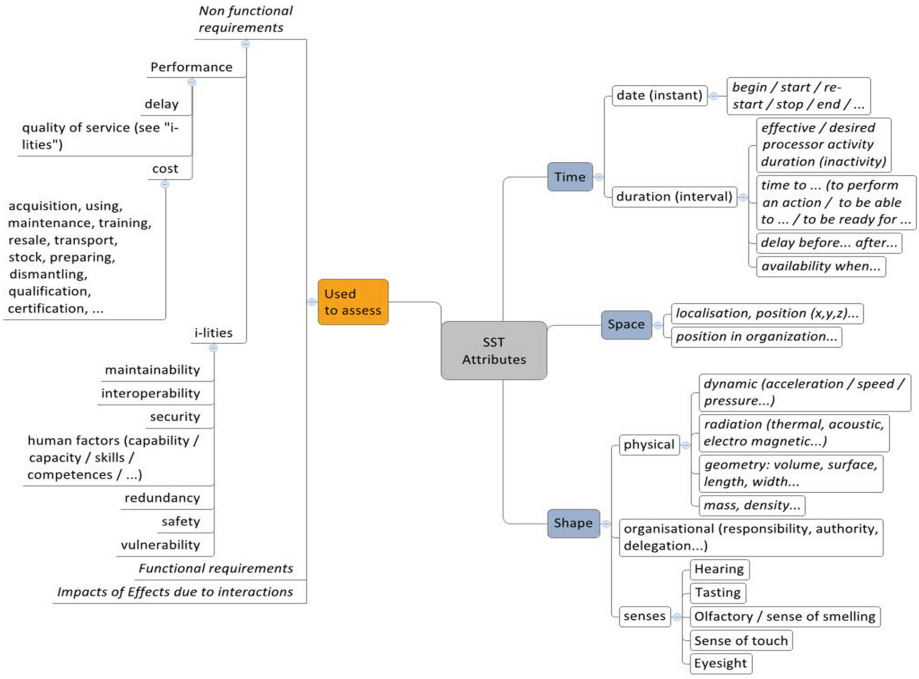

Fig. 2. SST attributes categories and examples in use

Analytical methods can be applied (1) to assess processor performance (e.g. in terms of costs, QoS or response time); and (2), to check some of the functional and/or non-functional **requirements** (e.g. by evaluating various "-*ilities*"[2]) the processor must respect in accordance with Stakeholders' expectations and constraints.

[2] « *developmental, operational and support requirements a program must address (e.g. availability, maintainability, vulnerability, reliability, or supportability)* » [15, 24] i.e. a kind of non-functional requirement (NFR).

An **interaction** is an oriented relation between an emitter processor P_1 and one or more receiver processors $\{P_2, \ldots, P_n\}$ denoted $\{P_i\}$. There is an interaction when (1) an exchange of one or more identified flows or service between P_1 and $\{P_i\}$ is identified and/or (2), one or more **fields** F generated by P_1 can impact $\{P_i\}$. An interaction is planned or desired, or alternatively unwanted or unintentional. In all cases, it can cause an **effect** if the interaction (1) affects one or more P_2 SST attributes, and/or (2), impacts the set of requirements (including interoperability requirements) to respect by $\{P_i\}$, in one of various characterized ways, that means [21]:

- Feared/Harmful. At least one characteristic of $\{P_i\}$ becomes inconsistent with the necessary conditions to survive. In this case, the identified relation causes the emergence of behaviours or physical phenomenon that are often inappropriate, such as resonances, electromagnetic interferences and thermal effects, thereby inducing rather risky situations (accident, incident, or malfunction) or damage to operational modes at the source and destination(s). They have to be avoided or simply modified.
- Required but absent. The effect should exist but remains absent for various reasons such as design mistakes or errors. In this case, some non-functional requirements concerning P_1 and $\{P_i\}$ have not been verified (performance, safety, security, etc.).
- Required and present. The effect exists and moreover is considered necessary. All requirements concerning P_1 and $\{P_i\}$ are checked so this effect cannot be removed or even modified.
- Required then appropriate but insufficient or excessive: The effect exists some non-functional requirements concerning $\{P_i\}$ are not checked yet (performance, safety, security, etc.). So, the effect must be analysed in order to be improved or reduced.

The **effect** can be derived from various dimensions, depending on the technical or socio-technical nature of processors P_1 and $\{P_i\}$. Figure 3 shows the proposed effect model inspired by the substance-field model originally proposed in [22]. In this model, a **field** F is from thermal, mechanical, pressure, biological or other nature. A list of available fields is given in [21] and [23] proposes a database of potential effects that can help designers to identify appropriate solutions for modifying the interaction.

After defining these concepts, S must respect [25, 26] stakeholders' requirements separated into functional (i.e. "*what must the system do?")* and non-functional ("*what are the system's expected characteristics of performance, "-ilities" and constraints supposed to do?*") requirements. In our case, interoperability might concern both functional and non-functional aspects of S.

The next formalizes the notion of interoperability requirement inspired by [27] (applied to collaborative processes).

These requirements are split up into 4 categories such as:

- **Compatibility.** S can send and receive flows from other systems in its environment whenever such interactions are needed. This ability is driven by respecting technical standards, communication protocols for technical compatibility or organizational rules and policies for organizational compatibility, described respectively as technical compatibility requirements e.g. required frequency of the exchange and organizational compatibility requirements. These must be recognized by all systems having to interact with S.

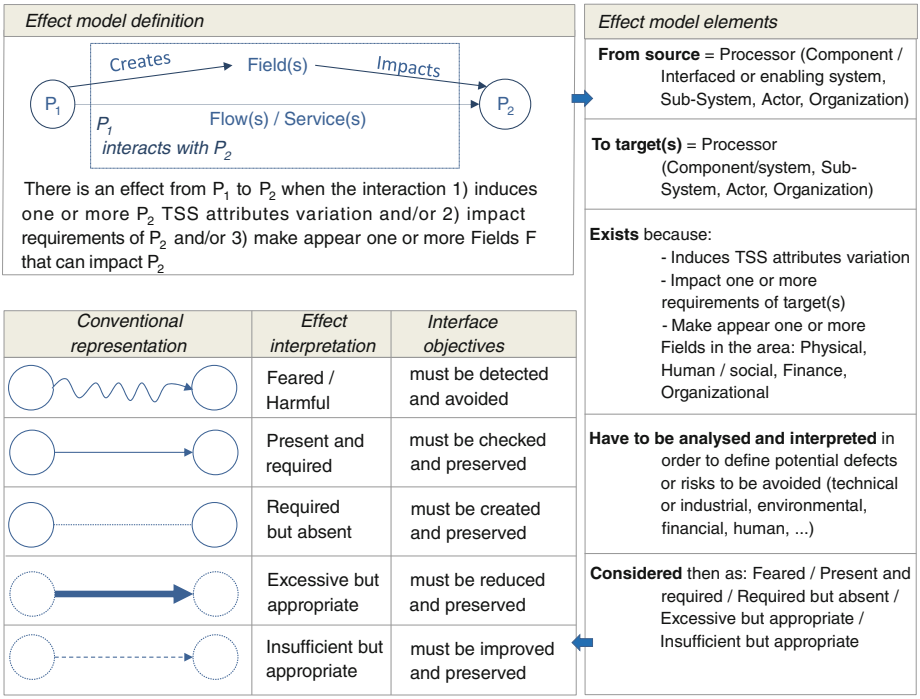

Fig. 3. Effect model principle

- **Inter-operation.** S operates seamlessly with the other systems in its environment by taking into account flows content being exchanged to fulfil its mission; moreover, it is able to control, adapt or anticipate problems promptly. S can also influence, not necessarily intentionally, other systems through both desirable and adverse effects. In this case, the term interoperation requirements could be referenced e.g. lifetime of any item transported before its obsolescence in taking into account states and modes of operation at the origin and relation target (ready, stop, etc.).
- **Autonomy.** S is independent of other system operations and behaviour. Autonomy may be decomposed into decisional autonomy (where S assumes its governance and remains capable of deciding actions) and operational autonomy (where S remains capable of preserving its performance in terms of cost, schedule and quality of service). At this point, it becomes necessary to consider decisional autonomy requirements and operational autonomy requirements.
- **Reversibility.** The relationships between S and the other systems are completely reversible, i.e. S can return to an identified configuration or state without causing any problems (dysfunctions, loss of performance, requirement violations, etc.) requiring difficult to manage changes once S no longer needs to exchange with the other systems in its environment. Relationship reversibility requirement is the term introduced here.

Last, a causality rule exists whereby: "*a processor A will be interoperable with processor B if all elements that compose (from different sources) or refine (from the same*

source) A and are involved in the interaction relationship with B do not cause interoperability problems, i.e. A and B respect interoperability requirements regarding their own role and objectives within the interaction".

Considering these new classes of requirements, requirements checklists classically used in industry can be then enriched as proposed in Fig. 4.

All concepts previously described are requested for designing interfaces as follows.

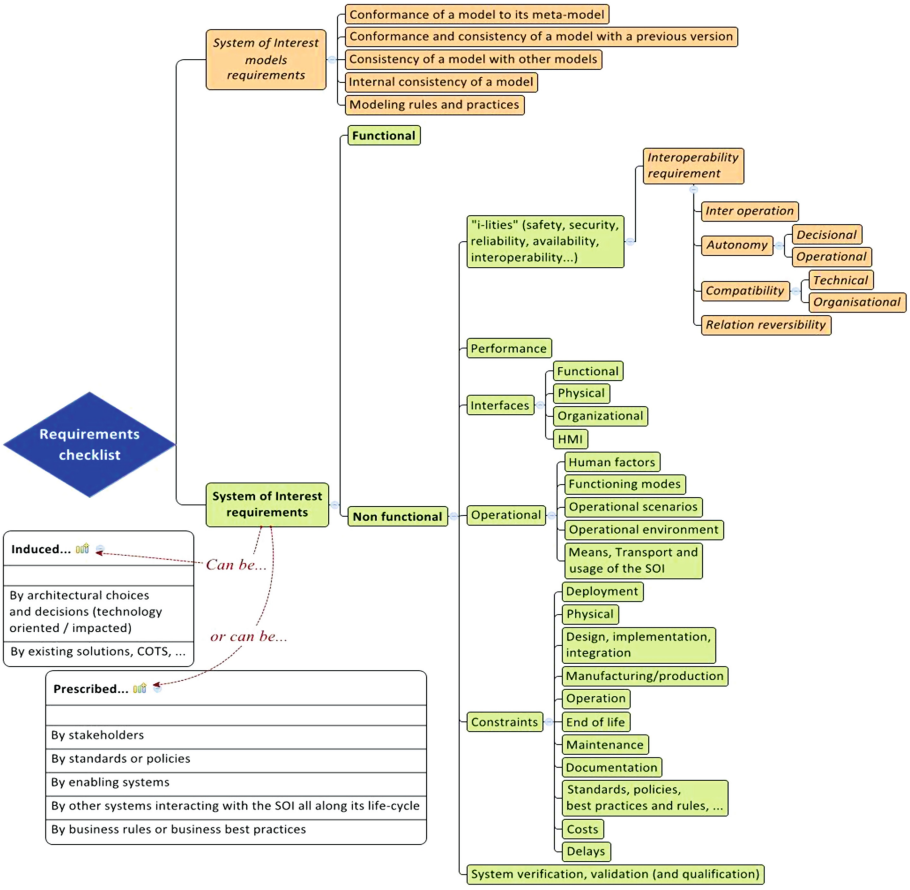

Fig. 4. Enriched requirements checklist

4 Interface Model Pattern

An interface is "*the common logical and/or physical border between two or more components* (here, processors) *or between the system* (a processor) *and its environment* (a upper-level processor)*, at which the rules of exchange, compatibility, integrity and non-regression are to be respected throughout the system's life cycle*".

From a theoretical point of view, an interface allows P_1 and each P_i $\{P_i\}$ to:

- Exchange the requested flow(s) or service(s). In due course, it must consider (in the receiver role) or make available (in the emitter role) the items carried out by the(se) flow(s) or requested by the service(s). Among other abilities, this set-up must:
  - Provide functional skills: emit, receive, transport, adapt (e.g. convert the input format of the exchanged flows), separate, protect, authorize the interaction, involve another processor, manage the items (e.g. store, retrieve), etc.
  - Respect all stakeholders' requirements, especially, interoperability requirements.
- Protect them from, or avoid, the potential effects induced by the interaction, or, at least, be able to contain the inappropriate effects within acceptable limits. To this purpose, its behaviour has to be adapted and/or a set of protection mechanisms or barriers must be designed to limit risky situations, for instance inspired by resilience engineering principles [28].

So we propose in the next an **interface pattern model** enabling engineers to model and analyse interactions between any type of systems and other systems composing the environment then to build and check interfaces.

An interface is conceptualized as a processor P intending to establish a connection between a processor P_1 (system, component, subsystem, business unit or actor) with its environment composed of a set of processors $\{P_i\}$ in order to (*objective 1*) transport flows between P_1 and $\{P_i\}$ (or vice versa) and/or (*objective 2*) protect from, in the sense of avoiding, unwanted effects between P_1 and (at least) one of the $\{P_i\}$ processors resulting from relationship implementation. Considering interface design and following system design principles (e.g. as proposed in [16, 29]), three cases must be raised:

- Designing a native interoperable processor P induces the design of each needed or potential interface, by considering each interface as a sub-processor of P.
- Improving the interoperability of an existing processor P can induce global or partial re-engineering of each of its interfaces, considering that each processor P1 found to play the role of interface can either replace one of the parts of P or be added to P.
- Improving the interoperability of processor P, by considering P impossible to modify (e.g. P must be definitively integrated into a more complex processor P', perhaps assumed to be the upper-level system). It induces the design and addition of new interfaces between P and the other processors from the environment.

In accordance with the basic principles of Systems Engineering approach, and as illustrated in Fig. 5 an interface can be viewed as a processor characterized by:

- **Interface Purpose:** The interface (objective 1) "*allow to ensure the exchange of flows or services between two or more systems (components/functions/actors or business units having to be identified), as expected from an efficient (in terms of resources used) and effective (with positive results) way*", or (objective 2) "*contributes to improve the protection of a system (to be identified) from an efficient and effective way, in taking into account other systems with which the interaction is not mandatory or even inevitable*".

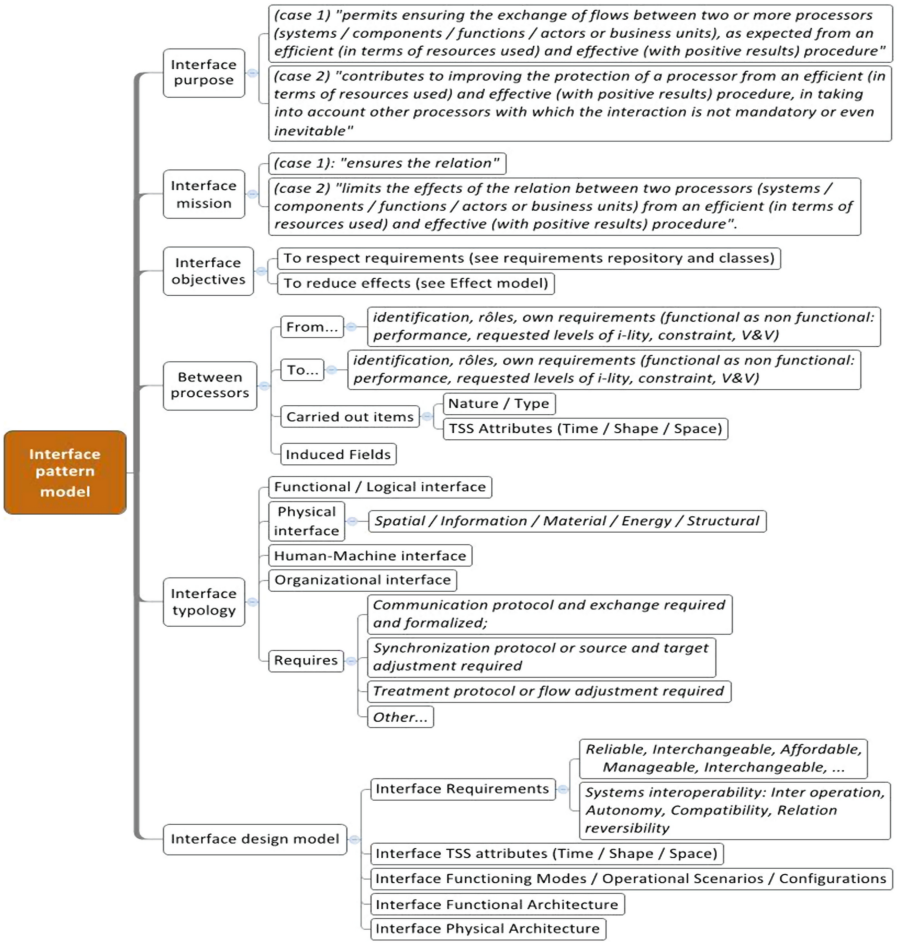

Fig. 5. Interface pattern model

- **Interface Mission:** An interface (objective 1): "*ensures the requested interaction i.e. exchange of flow(s), service(s) (themselves inducing exchange of flows)*" or (objective 2) "*limits the effects of the interaction between identified processors (systems/components/functions/actors or business units) from an efficient (in terms of resources used) and effective (with positive results) way*".
- **Interface Objectives:** An interface must respect overall functional and non-functional requirements (including interoperability but also, for instance, performance, ergonomics, constraints, or verification requirements). They are induced or come from the identified processors in relation. At least, an interface must improve the identified processors interoperability (i.e. compatibility, interoperation, autonomy and reversibility), reduce effects (having to be detected, identified and then modelled

by an effect model) following the interface objectives, interchangeable, reliable, affordable, scalable, manageable and interchangeable (Fig. 5).

- **Interface Typology:** logical or functional at first during the design process, it will become physical, human-machine (HMI) or organizational interface. So, it is proposed to distinguish the functional or logical interface, from physical, human-machine and organizational interfaces as follows:

- *Functional interface*: between functional entities (e.g. functions from functional architecture). The designer creates functional interfaces between functions that model the flows to be exchanged (data, material, energy) in the role of input, output, control (trigger) or resource flows. This notion of function requires determining:

  - types of carried out items, contents, origin (external of the system of interest or internal), and respective roles in the system;
  - whether or not a communication protocol and exchange is requested and formalized;
  - whether a treatment protocol or flow adjustment is needed taking into account interoperability requirements;
  - whether a synchronization protocol or source and target(s) adjustment is requested.

After allocating functions to the processors, the functional interface evolves into the physical, human-machine and/or organizational interfaces, which are then required between the system under design, and its context, or else between subsystems and components.

- *Physical interface*: between the system to be designed and components or subsystems forming its context [11, 29, 30]. These interfaces are required to:

  - Enable operating functions on physical flows and hence meeting the functional requirements. For instance, [11] distinguished five types of physical interfaces:
    - Spatial: related to physical adjacency for alignment, orientation, serviceability, assembly or weight;
    - Structural: related to load transfer or content;
    - Material: related to the transfer of airflow, oil, fuel or water;
    - Energy: related to the transfer of heat, vibration, electric or noise energy;
    - Information: related to the transfer of signals or controls;
  - Respect non-functional requirements (performance, "*-ilities*" such as interoperability when considering non-functional aspect of interoperability, and abilities, e.g. emission, reception, or transport of a flow);

- *Human-Machine interface* (HMI): The activities required for user interface design are already detailed for instance in [30].
- *Organizational interface*: These interfaces are required between actors and organizational units involved in and required to play roles in the system of interest. Exchanges become necessary in conducting sharing, collaboration, communication

and cooperation when performing activities to: produce/manufacture, deliver, store, sell, buy, design, manage, control, verify, plan, teach and organize training periods for stakeholders, qualify actors' profiles, decide, etc. These interfaces can be modelled as a collaborative working process model or a virtual organization model for instance.

- **Interface SST Attributes.** The goal is to define what are the requested SST attributes of the requested interface, for instance, in terms of potential physical elements that can be used to implement the interface (communication components, connections, ports, links, etc.) as illustrated in Fig. 2 and such as:
  - Time e.g. duration for connection,/disconnection, maximum delay before updating value or life cycle duration before obsolescence of the carried out items;
  - Shape e.g. dimensions (L*H*D), geometry, weight, radiation from various nature (see the list of possible fields in the effect model);
  - Space e.g. position, speed, transfer speed…
- **Interface Functioning Modes/Operational Scenarios and Configurations:** As any component, an interface evolves all along its life cycle by passing from a functioning mode to another one, highlighting then various behavioural scenarios and configurations. An approach for discovering and analysing these characteristics are detailed in [31].
- **Interface Functional Architecture:** The interface must transform one or more flows stemming from an emitter system to a (set of) receiver system. This transformation allows avoiding physical effects that may impair the systems in interaction (e.g. disturbing or damaging structure/organization or behaviour) and moreover must verify the interoperability requirements. We propose to model the expected transformation by (1) a model of time, shape and space attribute transformation of the flow and of items transported by the flow, and (2) the effect model proposed above focusing on the potential effects to be avoided and anticipated. In design stage, the functional vision of an interface highlighting these two transformations can be for instance modelled by choosing and using one of the modelling languages introduced in Fig. 1 considering the nature of source and targets processors. This completes the interface pattern model with one or more functional architecture patterns models, more or less detailed aiming facilitating design by reusing partially or fully these models.
- **Interface Physical Architecture:** The interface is implemented by linking various sub-processors (physical subsystems or components, actors, sub-organizational units), on which the functions proposed in the functional architectures are to be allocated et then performed taking particularly into account all non-functional requirements. This description can be generated by using, for instance, any Physical Block modelling language and respecting SE principles.

5 Conclusion and Prospects

This paper has introduced conceptual aspects of an interface model pattern supporting engineers involved in natively interoperable system's design activities. This helps particularly and guides modelling activities but aims also to permit checking and testing conformity, coherence and adequacy [32] of proposed interfaces in order to design a system that will be able to maximise its interoperability in various situations even difficult to predict. The goal is now to develop modelling and analysis platform [33] integrating existing proof and simulation tools [34, 35] allowing then mixing formal properties proof and simulation as proposed in [36] when considering systems of systems [37] interoperability analysis.

References

1. Camarinha-Matos, L.M.: Collaborative networks: a mechanism for enterprise agility and resilience. In: Mertins, K., Bénaben, F., Poler, R., Bourrières, J.-P. (eds.) Enterprise Interoperability VI, pp. 1–8. Springer International Publishing, Switzerland (2014)
2. European Commission, Annex II - EIF (European Interoperability Framework) (2010). http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf. Accessed 20 February 2015
3. ATHENA Interoperability Framework (AIF). http://athena.modelbased.net/model.html. Accessed 20 February 2015
4. Department of Defense (DoD) Architecture Framework (DoDAF), version 2007, 23 April, version 1.5 (Volumes I, II and III) (updated version 2.02 is available)
5. Daclin, N., Chen, D., Vallespir, B.: Methodology for enterprise interoperability. In: Proceedings of the 17th World Congress - The International Federation of Automatic Control - Seoul, Korea, 6–11 July 2008
6. Lavean, G.: Interoperability in defense communications. Commun. IEEE Trans. **28**, 1445–1455 (1980)
7. Clark, T., Jones, R.: Organisational interoperability maturity model for C2. In: 1999 Command and Control Research Technology Symposium. United States Naval War College, Newport (1999)
8. De Soria, I.M., Alonso, J., Orue-Echevarria, L., Vergara, M.: Developing an enterprise collaboration maturity model: research challenges and future directions. In: 15th International Conference on Concurrent Enterprising, Leiden, Netherlands, 22–24 June 2009
9. Naudet, Y., Latour, T., Chen, D.: A systemic approach to interoperability formalization. In: Proceedings of the 17th World Congress IFAC, International Federation of Automatic Control, Seoul, Korea, 6–11 July 2008
10. DoD, Systems Engineering Fundamentals. Defence Acquisition University Press (2001). http://www.dau.mil/pubscats/PubsCats/SEFGuide2001-01.pdf. Accessed 7 July 2012
11. AFIS, CT AIS, Interfaces techniques et architectures du système, fiche N°2 (2006) (in French)
12. Le Moigne, C.: La théorie du système général: théorie de la modélisation (1994) (in French)
13. Jean-Michel, P.: La modélisation par les systèmes en situations complexes. Thèse de Doctorat, Université de Paris Sud, France (1997) (in French)

14. Féliot, C.: Toward a formal theory of systems, Colloque d'Automne du LIX 2007 - CAL07 Complex Systems: Modelling, Verification and Optimization, Paris, Carré des Sciences, 3rd and 4th October 2007, http://www.lix.polytechnique.fr/∼liberti/cal07/presentations/. Accessed 11 July 2012
15. INCOSE, System Engineering (SE) Handbook, A Guide For System Life Cycle Processes And Activities Version 3.2.2, INCOSE TP 2003 002 03.2.2 (2011)
16. BKCASE Editorial Board. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.3. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology (2014). http://www.sebokwiki.org/. Accessed 17 July 2014
17. Blanchard, B.S., Fabricky, W.J.: Systems Engineering and Analysis. Prentice Hall International Series in industrial and systems engineering, 5th edn. Pearson College, London (2011)
18. Charlotte Seidner, Vérification des EFFBDs : Model checking en Ingénierie Système, Ph.D. Nantes University, 3 November 2009 (in French)
19. OMG, Business Process Modelling Notation 2.0. http://www.omg.org/spec/BPMN/2.0/PDF/. Accessed 16 March 2015
20. System Modeling Language SysML. http://www.sysml.org/. Accessed 16 July 2014
21. Mann, D.: Hands on Systemic Innovation. CREAX Press, Belgium (2002)
22. Altshuller, G.: The TRIZ method: numerous references and a presentation of TRIZ principles. http://www.altshuller.ru/world/eng/index.asp. Accessed 17 July 2012
23. Oxford creativity, Effects data base. http://wbam2244.dns-systems.net//EDB_Welcome.php. Accessed 27 November 2012
24. de Weck, O.L., Ross, A.M., Rhodes, D.H.: Investigating relationships and semantic sets amongst system lifecycle properties (Ilities). In: Third International Engineering Systems Symposium CESUN 2012, Delft University of Technology, 18–20 June 2012
25. INCOSE, Survey of Model-Based Systems Engineering (MBSE) Methodologies, INCOSE-TD-2007-003-01, Version/Revision: B, 10 June 2008
26. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: ICSE 2000, Proceedings, Conference on the Future of Software Engineering, New York (2000)
27. Mallek, S., Daclin, N., Chapurlat, V.: An approach for interoperability requirements specification and verification. In: van Sinderen, M., Johnson, P. (eds.) IWEI 2011. LNBIP, vol. 76, pp. 89–102. Springer, Heidelberg (2011)
28. Hollnagel, E., Paries, J., Woods, D., Wreathall, J. (eds.): Resilience Engineering in Practice: A Guidebook. ASHGATE Publishing Company, Farnham (2011). ISBN: 978-0-4094-1035-5
29. Thimbleby, H., Blandford, A., Cairns, P., Curzon, P., Jones, M.: User interface design as systems design. In: Faulkner, X., Finlay, J., Detienne, F. (eds.) People and Computers XVI-Memorable Yet Invisible, Proceedings of HCI 2002. Springer (2002). ISBN: 1852336595
30. Gruhn, P.: Human machine interface (HMI) design: the good, the bad, and the ugly (and what makes them so). In: 66th Annual Instrumentation Symposium for the Process Industries, 27–29 January 2011
31. Chapurlat, V., Daclin, N.: Proposition of a guide for investigating, modeling and analyzing system operating modes: OMAG. In: International Conference on Complex System Design and Management CSDM 2013, Paris, December 2013
32. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., McKenzie, P.: Systems and Software Verification: Model Checking Techniques and Tools. Springer, Heidelberg (2001)
33. Nastov, B., Chapurlat, V., Dony, C., Pfister, F.: A verification approach from MDE applied to model based system engineering: xeFFBD dynamic semantic. In: Internation Conference on Complex System Design and Management CSDM 2014, Paris, France, December 2014

34. Formal verification tools overview web site. http://anna.fi.muni.cz/yahoda/. Accessed 10 April 2011
35. V&V Tools, RPG reference document (2006). http://vva.msco.mil/Ref_Docs/VVTools/vvtools-pr.PDF. Accessed 10 April 2011
36. Bilal, M., Daclin, N., Chapurlat, V.: System of systems design verification: problematic, trends and opportunities. In: Mertins, K., Bénaben, F., Poler, R., Bourrières, J.-P. (eds.) Enterprise Interoperability VI, pp. 405–415. Springer International Publishing, Switzerland (2014)
37. Ferris, T.L.J.: It Depends: Systems of systems engineering requires new methods if you are talking about new kinds of systems of systems, INCOSE (2006)