The application of interoperability requirement specification and verification to collaborative processes in industry

Sihem Mallek, Nicolas Daclin, Vincent Chapurlat

The application of interoperability requirement specification and verification to collaborative processes in industry

S. Mallek, N. Daclin *, V. Chapurlat

LGI2P – Laboratoire de Génie Informatique et d'Ingénierie de Production, Site de l'Ecole des Mines d'Alès, Parc Scientifique G. Besse, 30035 Nîmes cedex 1, France

A B S T R A C T

Interoperability is becoming a crucial issue for industry, and a lack of interoperability can be seen as an important barrier to collaborative work, in both public (inter-enterprise) and private (intra-enterprise) collaborative processes. Indeed, interoperability is generally defined as the ability of enterprises to interact within a collaborative process. Prior to any effective collaboration, it is necessary to inform enterprises, which aim to work together, whether or not they would be able to interoperate. Research on interoperability has shown the benefits of measuring and evaluating interoperability, by using several frameworks and maturity models. However, approaches for detecting and anticipating interoperability problems do not seem to exist. Our research proposes to use formal verification techniques to detect different types of interoperability problems. On the one hand, this means being able to define the particular interoperability needs to be considered. On the other hand, it requires these needs to be formalized as a set of unambiguous and, as formally stated as possible, requirements. Moreover, interoperability requirements can have temporal or a-temporal features. To detect interoperability problems in anticipative way, interoperability requirements must be checked by means of a target process model. Three complementary verification techniques are used to verify interoperability requirements in a collaborative process model. The verification technique used depends on the aspect and the level of abstraction of the requirement to be verified. This paper focuses and illustrates the detection of interoperability problems using verification techniques.

Keywords:
Interoperability requirements
Verification
Collaborative process

1. Introduction

In recent decades, enterprises have developed and applied various principles and organizational systems to improve collaboration with external partners (i.e., inter-enterprise collaboration), and collaboration between internal teams (i.e., intra-enterprise collaboration). These approaches have allowed them to meet the challenges created by the globalization of markets, and especially to be able to focus on their own domain of expertise with no loss of performance, and to share experiences, processes, and tools with internal or external partners confidently. In this context, it is important to describe and formalize how different partners (e.g., an isolated player, team, department, or enterprise) can work with others and through their interactions achieve a common objective. This formalization is performed using the principles of collaborative processes, which can be public (“process activities belong to different organizations” [2]) or private (“set of activities ordered in a set of procedural rules to achieve a specific goal within an organization and carried out by a group of persons” [3]) [1]. The ability of partners to interact is one of the key factors to characterize and assess, in order to measure the overall performance of a collaborative process. This factor is related to the concept of interoperability which can be defined as the “ability of enterprises and entities within those enterprises to communicate and interact effectively” [4]. Therefore, enterprises involved in collaborative processes have to improve their interoperability. In order to accomplish this goal, they must be able to detect, anticipate, and solve their interoperability problems. Our research addresses the issue of detection from an anticipative point of view – i.e., before the implementation of the collaborative process – attempting to detect problems of interoperability that can be induced by the characteristics and behaviors of partners. In this perspective, in order to anticipate problems, analysis of the collaborative process model must first be performed. Second, the anticipation of problems is performed according to the satisfaction of interoperability needs by the collaborative process model. In this way, to demonstrate that a need is satisfied, it must first be formalized into a requirement and various verification techniques must be applied to the collaborative process model. Based on these considerations, our research aims, first, to define, structure, and formalize interoperability requirements that partners must satisfy. Second, it aims to enhance and implement a set of verification

* Corresponding author. Tel.: +33 466 387 066.
E-mail address: surname.name@mines-ales.fr (N. Daclin).

	Needs	Measurement	Formalization	Verification	Local detection	Operational aspects
Maturity models	+++	-	-	-	-	-
Interoperability measurement	-	+++	-	-	-	++
Operational effectiveness	-	+++	-	-	-	+++

Fig. 1. Comparative study of the different research work.

techniques that can be used prior to implementing any steps in the collaborative process.

This paper focuses on the specification and verification of interoperability requirements within collaborative processes, and is organized as follow. Section 2 reviews several research projects focusing on the interoperability analysis. Section 3 presents the different steps of the proposed approach to detect interoperability problems. Section 4 explains how interoperability needs are obtained. Section 5 proposes the definitions and a classification of interoperability requirements. Section 6 presents the analysis of interoperability requirements using complementary verification techniques. The seventh section gives a case study in order to illustrate the verification of interoperability requirements.

2. Interoperability analysis: existing research and discussion

The development of interoperability has become a crucial question for enterprises that want to become more competitive in a globalized environment. As a consequence, a large body of research has been developed in recent years in order to analyze interoperability (i.e., analysis that formalizes, evaluates, and measures system interoperability). Concerning formalization, interoperability frameworks (e.g., IDEAS, INTEROP, AIF...) [5–7] structure interoperability requirements according to different aspects such as interoperability problems that can occur or level at which interoperability takes place. For instance, the INTEROP framework considers two fundamental aspects: the "interoperability barriers" (conceptual, organizational and technological) and "interoperability concerns" (data, services, processes and business). The goal is to achieve efficient collaboration by overcoming interoperability barriers relating to interoperability concerns. Although these frameworks can be used to create a better classification and structuring of the basic aspects of interoperability, they cannot be used to evaluate interoperability. The evaluation of interoperability is mostly performed with maturity models (LISI, LCIM, OIM...) [8–10], which can be used to characterize the ability of systems to interoperate at a given moment with regards to different identified interoperability levels. Some of these models give recommendations to go from one interoperability level to a higher interoperability level. However, these maturity models focus on the qualitative evaluation of interoperability, and do not offer a quantitative measurement of interoperability. A methodology has been proposed to measure interoperability of enterprises in both intra- and inter-enterprise contexts [11]. This methodology measures compatibility and interoperation performance (i.e., the dynamic aspect of the collaboration) simultaneously. To measure compatibility, a compatibility matrix is used to inform enterprises as to whether or not) of interoperability problems are expected to occur. To measure interoperation performance, some criteria such as cost of interoperation, time of interoperation, and quality of interoperation are measured. Similarly, the research proposed by [12] is based on the development of several modes of interoperability aiming to offer operational effectiveness. However, this research does not formalize interoperability so as to measure it, and cannot be used to detect and identify local interoperability problems in a formal way. Fig. 1 provides a comparative overview of the research presented above.[1] It shows the six main characteristics that have inspired our research work, which include (1) the expression of interoperability needs, (2) the measurement of interoperability, (3) the formalization of interoperability, (4) the use of verification techniques, (5) the local detection of interoperability requirements, and finally (6), the operational aspects of the collaborative process.

These different research studies have evaluated or provided an indication as to whether or not there are interoperability problems, but they do not specifically describe these problems or give a formal proof of their existence. In addition, many gaps exist, such as (1) a lack of interoperability needs and requirements formalization, (2) a lack of collaborative process analysis tools, (3) a lack of detection principles for problems in collaborative process models, which would enable them to be anticipated, and finally (4) a lack of consideration of the operational aspects of the collaborative process. These gaps are ultimately related to the level of formalization, the problems to be detected, and the concepts needed to detect these problems. They are also related to techniques that can provide the irrefutable proof of the existence of interoperability problems formally (not only qualitatively or quantitatively).

3. Detection of interoperability problems: our approach

To detect interoperability problems in collaborative processes in an anticipative way, we propose a model-based approach for interoperability requirements checking. Our approach is inspired by requirements engineering that can be used to describe and structure interoperability requirements that are related to any interoperability problem that may hinder a collaborative process. Finally, this approach is based, on the one hand, on the use of formalization, and on the other hand, on the use of various verification techniques to ensure that the collaborative process model complies with these interoperability requirements. Thus, the local detection of interoperability problems is performed by the use of modeling, formalization, and verification techniques. This approach is implemented following the steps shown in Fig. 2.

First, interoperability needs are collected and expressed using a language close to the one used by the stakeholders involved in the implementation, execution, and control of a given intra- or inter-enterprise collaborative process. The objective of this step is to develop a repository of interoperability needs that is as comprehensive and generic as possible.

These needs are then clarified, structured, and formulated as the so-called interoperability requirements described below, then gathered into a repository of interoperability requirements.

[1] +++: best address the issue, ++: partly address the issue, +: relevant to the issue, −: irrelevant to the issue.

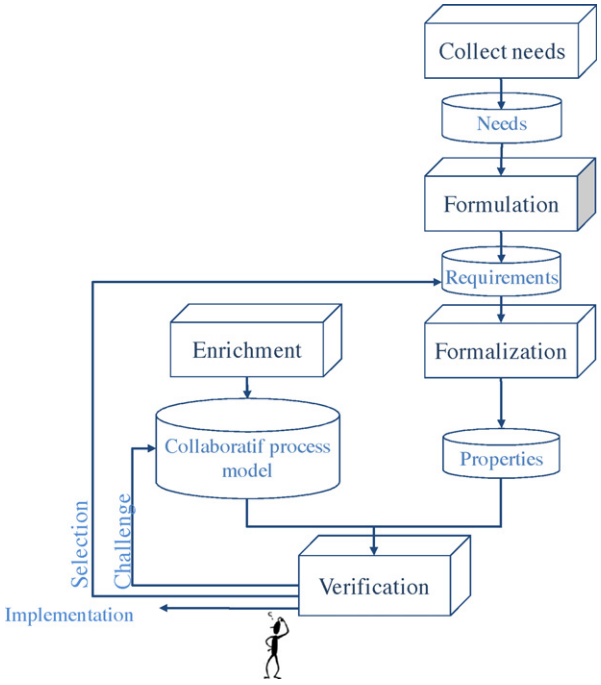


Fig. 2. The proposed approach.

Formulating and structuring the requirements allow the user in charge of the collaborative process model analysis to select the appropriate interoperability requirements to be checked, according to analysis objectives. This may include, for example, interest in process performance or in the compatibility of the tools used in the collaborative process. Once these interoperability requirements have been selected, they are formalized into properties through the use of a formal language so that they can be checked on the model with existing formal verification techniques. From this perspective, the collaborative process model must first consider the interoperability requirements. Thus, the collaborative process is modeled with an enriched version of BPMN (Business Process Modeling Notation) [14] in order to integrate the concepts related to interoperability so as to make the verification of interoperability requirements possible. Then, the model must be translated into a formal structure supporting the proposed verification technique, which requires model transformation mechanisms. Finally, the result of the verification indicates whether or not a requirement is really satisfied by the collaborative process model, which in turn points to the existence of an interoperability problem. Thus, if all selected requirements are satisfied by the collaborative process model, it can be implemented. If not, (1) the collaborative process model can be challenged to provide adequate solutions to the problems detected, and (2) the user can select other requirements before performing other verifications. These steps are explained in detail in the following sections.

4. Establishing interoperability needs

The first step aims to create a list of interoperability needs. The review of the state of the art presented in Section 2, which covers the different work related to interoperability, identified several interoperability needs. For example, the levels proposed in maturity models, to achieve full interoperability, express many interoperability needs. These maturity models cover the three main aspects of interoperability (conceptual, organizational and technological), and express needs related to these aspects such as:

- Need 1: "*a communication protocol exists for exchanging data between participating systems*" (LCIM – level 1) [9].
- Need 2: "recognized frameworks are in place to support interoperability and shared goals are recognized and roles and responsibilities are allocated as part of on-going responsibilities, however, the organizations are still distinct" (OIM – level 2) [10].
- Need 3: "*The connection systems are identified by their ability to provide an understanding of the data exchanged. The emails include attachments and contribute to the success of trade*" (LISI – level 2) [8].
- Need 4: "*The governance of the procurement process explicitly links the business requirements to technical architecture through project financing*" (IMM – level 4) [15].

To consolidate this list of needs and to enrich it, an industrial survey was conducted. The purpose of this survey was to give partners the opportunity to describe problems they face in terms of interoperability and to express them. This questionnaire focuses on several aspects. The first aspect highlights the need for collaboration within and between partners, which can be used to differentiate between the needs in a private collaborative process and a public collaborative process. The second aspect focuses on interoperability problems (conceptual, organizational and technological). Finally, the performance aspects are considered using standard criteria: cost, time, and quality. The requirements listed below are extracted from the answers provided by the industrial survey:

- Need 5: "*We need to send and receive exploitable data, and be sure that they are received by the other partner during the activity.*"
- Need 6: "*Homogenize communication whatever the form, then have a semantic understood and shared by both partners.*"
- Need 7: "*Keep the performance as good as possible in terms of cost, quality (product and service) and time spent, after the partnership is over, as before it was started.*"
- Need 8: "*A resource must be available when it is needed.*"

The needs collected in the bibliographic analysis and the needs collected in the survey are both expressed in natural language. They remain difficult to handle due to their nature and the complexity of their expression. Therefore, repetition, ambiguity, imprecision and incoherence must be removed. To address these problems mainly related to the expressiveness of natural language, these needs are formulated and structured in the form of requirements as proposed by requirements engineering [16,17].

5. Interoperability requirements: classification and structuring

A requirement is defined as: "*a statement that specifies a function, ability or a characteristic that a product or a system must satisfy in a given context*" [18]. Interoperability requirements are formulated thanks to the repository of interoperability needs described in the previous section. These requirements must be clearly expressed, identifiable, traceable, verifiable, unambiguous, and not contradict another requirement from the same repository. All requirements obtained are placed in a repository of interoperability requirements. This repository is used as a reference repository to classify and structure interoperability requirements.

5.1. Classification of interoperability requirements

In general, interoperability is related to the concept of compatibility. Compatibility means to harmonize enterprises in

terms of method, organization, and tools, so that the heterogeneous information exchanged can be understood and exploited by each of them with no extra interfacing efforts. However, compatibility is not sufficient to describe interoperability. In fact, during a collaborative process, various other sources of problems can be distinguished. For instance, two enterprises use the same language to code their data, but one of them is unable to send its data. In this case, "*two interoperable systems are compatible, but two compatible systems are not necessarily interoperable*" [19]. This statement leads to the notion that interoperability is related to interoperation during collaboration. Interoperation focuses on real interactions between enterprises during collaboration (i.e., it concerns the ability of enterprises to work efficiently together throughout the interactive process and take into account various events that may occur). Furthermore, interoperability is related to the preservation of autonomy during collaboration, which means that partners can work together (e.g., exchange services) while continuing to follow their own logic of operation [6]. Moreover, when collaboration stops, enterprises wish to return to their previous performance level while remaining efficient. For instance, one partner may note a loss of performance with regards to its performance before the collaboration. In this case, it is question of being sure that a given partner will be able to recover his or her own performance level after the end of the collaboration. Thus, interoperability is related to another concept called reversibility [6]. Reversibility means that partners can return to their original state at the end of collaboration with regards to their performance and including positive and/or negative variation accepted prior to any real collaboration.

In consideration of the aforementioned, an interoperability requirement is defined as: "*a statement that specifies a function, ability or characteristic, related to the ability of a partner to ensure its partnership in terms of compatibility*, *interoperation*, autonomy, and *reversibility, that it must satisfy*". Thus, four categories of interoperability requirements are defined: compatibility, interoperation, autonomy, and reversibility.

A compatibility requirement is defined as "a statement that specifies a function, ability, or characteristic, considered to be invariable throughout the collaboration and related to interoperability barriers (conceptual, organizational, and technological) for each interoperability concern (data, services, processes, and business), and which partners must satisfy before collaboration is effective".

Compatibility requirements focus on the interface between the partners involved. All of these requirements must be checked and are invariable throughout the duration of the collaboration. For instance, three compatibility requirements can be formulated from the fifth need presented in Section 4.

- $CDT_{Translation}$[2]: "*The receiving partner has the necessary means to translate the data exchanged*". This CR is related to technological aspects of interoperability at the level of data. It reflects the needs of the receiving partner in terms of the translation of homogeneous or heterogeneous data, and subsequently their exploitation.
- $CDO_{Authorization}$: "*The sending partner has the required permissions to carry out exchanges with the receiving partner*". This CR is related to organizational issues and reflects the needs of partners to exchange their data safely.
- $CSO_{Manager}$: "*Each service has a clearly defined manager with authority over the rest of the team*". This CR is expressed at the level of services and is related to the organizational aspect. The objective of the implementation of this requirement is to avoid, for example, loss of time which may be harmful to the process during its execution.

An interoperation requirement is defined as "a statement that specifies a function, ability or characteristic, *considered to be variable during the collaboration*, related to the performance of the interaction, and which each partner must satisfy".

Interoperation requirements are related to the execution phase of the collaborative process. In addition, their veracity can be considered to be variable during the collaboration with regards to the interactions themselves. For example, some interoperation requirements can be expressed from the first and eighth need.

- $ISO_{Receipt}$: "*The receiver sends a receipt to the sender*". Formulated from need number 1, this IR is related to the organizational aspect at the level of services.
- $ISO_{AvailabilityTime}$: "*Resource is available when an activity needs it*". Formulated from need number 8, this IR focuses on the availability of the resource at the moment it is requested by and for an activity.
- $ISO_{ExecutionTime}$: "*Resource is active throughout the duration of execution of the activity*". This IR is formulated to ensure that a resource can really be exploited while a given activity is being executed. The goal is to ensure that this resource will not be allocated to another activity for the duration of the first activity that uses it.

An autonomy requirement is defined as "a statement that specifies a function, ability or characteristic *related to the ability of partners to perform their own governance and maintain their own operational capacity during collaboration*, and which each partner must satisfy".

Autonomy requirements are related to the ability of partners to maintain their independence while they are involved in a collaborative process. This is characterized by self-governance and operational autonomy. Self-governance is necessary for the partners to keep their freedom in their decision without conflict and disagreements that can hinder the collaboration. Operational autonomy is necessary so the partners can maintain a certain degree of freedom in their actions to achieve their own goals. Both types of autonomy must be present at the business, process, and service level of the enterprise and for each interoperability barrier (conceptual, technological, and organizational). For instance, from need number 4 expressed above, three autonomy requirements can be formulated.

- $ABO_{Governance}$: "*Partner involved in a collaborative process is responsible for selecting its suppliers*" and $ABO_{GovernanceCost}$: "Partner *control costs related to the choice of suppliers*". These two requirements are expressed at the business level and for the organizational barrier. They ensure that each partner has the choice of its suppliers and associated costs.
- $APT_{Operational}$: "*A process must keep required physical flow of supply*". This AR concerns the particular operational autonomy in a process and the technological barriers.

A reversibility requirement is defined as "a statement that specifies a function, ability or characteristic, related to the capacity of a partner to go back to its original state (in terms of performance) after collaboration, and which each partner must satisfy".

Reversibility requirements are related to the capacity of a given system to return to its initial state even if the implementation of the collaborative process leads to various adaptations and/or changes (e.g. organization, working method, new tool, etc.). In fact,

[2] Interoperability requirements are expressed with the following conventions:
- First letter: class of interoperability
- Second letter: interoperability concern
- Third letter: interoperability barrier
- Index: name of the requirement.

at the end of the collaboration, each partner must be able to continue to further its own purpose, its mission, and achieve its objectives. This mainly concerns the impact on the integrity, stability, and performance of the system. With regard to integrity, the system must return to a known operating mode. Regarding stability, the system must be able to maintain its viability. Regarding performance, the system must recover the level of performance that characterized it before the collaboration including variations (i.e., the system may have to accept a loss of performance that is known and defined in advance. To give an example, it is possible to extract three reversibility requirements from need number 7, which focus on performance criteria.

- $RST_{PerformanceCost}$: "The cost of an activity after the collaboration corresponds to the cost before the collaboration including variations". This requirement concerns the cost criterion.
- $RST_{PerformanceTime}$: "The execution time of an activity after the collaboration corresponds to the execution time before the collaboration including variations". This requirement is related to the time criterion.
- $RST_{PerformanceQuality}$: "The number *of products that* are *provided by an activity after collaboration matches – including tolerances – the number of products provided before the collaboration*". This requirement concerns the performance quality criterion.

Finally, a Graph of Decomposition of Interoperability Requirements (GRADEI) is proposed in order to facilitate the description, handling, and decomposition of an interoperability requirement.

5.2. The GRADEI model: structure and propagation

The GRADEI model is a graph used to formally represent the requirements repository. It is based on (1) the principle of a decomposition relationship between requirements, (2) the previously proposed classification of interoperability requirements, and (3) the interoperability framework developed within the INTEROP Network of Excellence [6]. It helps the user to organize, reuse, and select the appropriate interoperability requirements with regards to a given collaborative process for the purpose of checking. GRADEI is an oriented graph as presented in Fig. 3.

A node represents a requirement at a given level of detail. An arc linking a node at one level of detail to other nodes at a lower level of detail represents the decomposition relationship. This relationship, which links a target node (i.e., representing an abstract requirement at level k) to its source nodes (i.e., representing more precise requirements at level $k+1$) is constrained by a logical function. This function expresses the influence of the satisfaction of requirements represented by the source nodes on the requirement represented by the target node. The following notations are used to formalize the GRADEI model:

- p = number of levels of detail, $p \in \mathcal{N}$.
- m = number of nodes of the graph, $m \in \mathcal{N}$.
- n = number of arcs of the graph, $n \in \mathcal{N}$.
- Π: set of moments (times).
- String:: = ('a'…'z'|'A'…'Z') ('a'…'z'|'A'…'Z'|'1'…'9'|'_'|'-')}
- Fact = {fact|fact ∈ VM ∪ PM ∪ Pr} where:
  - VM = {variables extracted from the process model}
  - PM = {parameters extracted from the process model}
  - Pr = {predicates extracted from the meta model of enriched BPMN}

GRADEI is formalized with a graph G:: = (A, N, N_0) where:

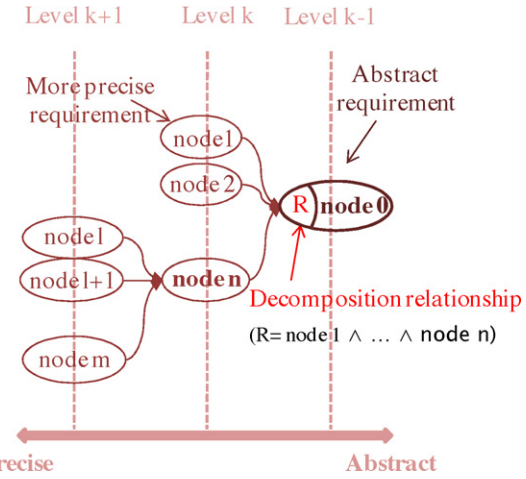


Fig. 3. Decomposition relationship between requirements in the GRADEI model.

- A represents the set of arcs linking nodes: $A = \{A_j / j \in [0,n]$; A_j:: = ($SourceN$, $TargetN$) with ($SourceN$, $TargetN$) $\in N \times N$, $SourceN \neq TargetN$ and $level(SourceN) >$ level($TargetN$)}
- N represents nodes of the graph: $N = \{N_i / i \in [0,m]$; N_i:: = ($name_i$, $description_i$, $relationship_i$, $fact_i$, $level_i$, $value_i$)} where:
  - ($name_i$, $description_i$) ∈ string × string
  - relation$_i$:: = (T, θ_e, θ_c) with:
- $T \subseteq \Pi$
- θ_e: $fact_i \cup T \rightarrow \{0, 1\}$ is the logical function that describes the necessary but not sufficient condition in which the node N_i is verified. This condition is expressed only in the modeling variables, parameters, and modeling predicates taken from the collaborative process model.
- θ_c: $NN_i \rightarrow \{0,1\}$, i.e. $\{value(N_1),\ldots,$ value $(N_k)\} \rightarrow \{0,1\}$ is the logical function that describes the necessary but not sufficient condition where the requirement represented by the node N_i is verified. This condition can be used to interpret the results (values) of nodes sources. NN_i is the set of node N_i sources defined as follows:
- $NN_i = \{N_j / \exists A_k(N_j, N_i), k \in [0,n], j \in [0,m]$ with $i \neq j\}$
  - $fact_i \in$ Fact.
  - $level_i \in [0; p[$ indicates the level of detail of the requirement. By definition, the root element has level = 0, i.e. interoperability in this context and level(N_i) returns the level$_i$ of node N_i.
  - $value_i \in \{0,1\}$ shows the verification result, *in absentia* 0 (false) and where $value_i = \theta_e \wedge \theta_c$.
- $\exists!\ N_0 = (name_0, description_0, relation_0, fact_0, level_0 = 0, value_0) \in N$ is the root node of the graph G representing the most abstract interoperability requirement.

It is worth noting that a requirement can be characterized by a temporal aspect. It can be a-temporal, that is to say independent of time. In this case, it is verifiable at all times (the set T is empty). It can be temporal, in other words, verifiable only at certain stages in the collaboration life cycle. The temporal aspect of a requirement is an important element for choosing the appropriate verification technique. Furthermore, the choice of the logical function (θ_c) used to link an abstract requirement to more precise requirements is left to the user's discretion. Thus, the requirements of each level can be analyzed separately. In this case, some of the requirements that are not satisfied, can be considered to be negligible or as representing an acceptable risk for the user. The interoperability requirements reference repository obtained is illustrated in Fig. 4.

As an example, for the GRADEI model presented Fig. 5, propagation of interoperability requirements analysis is

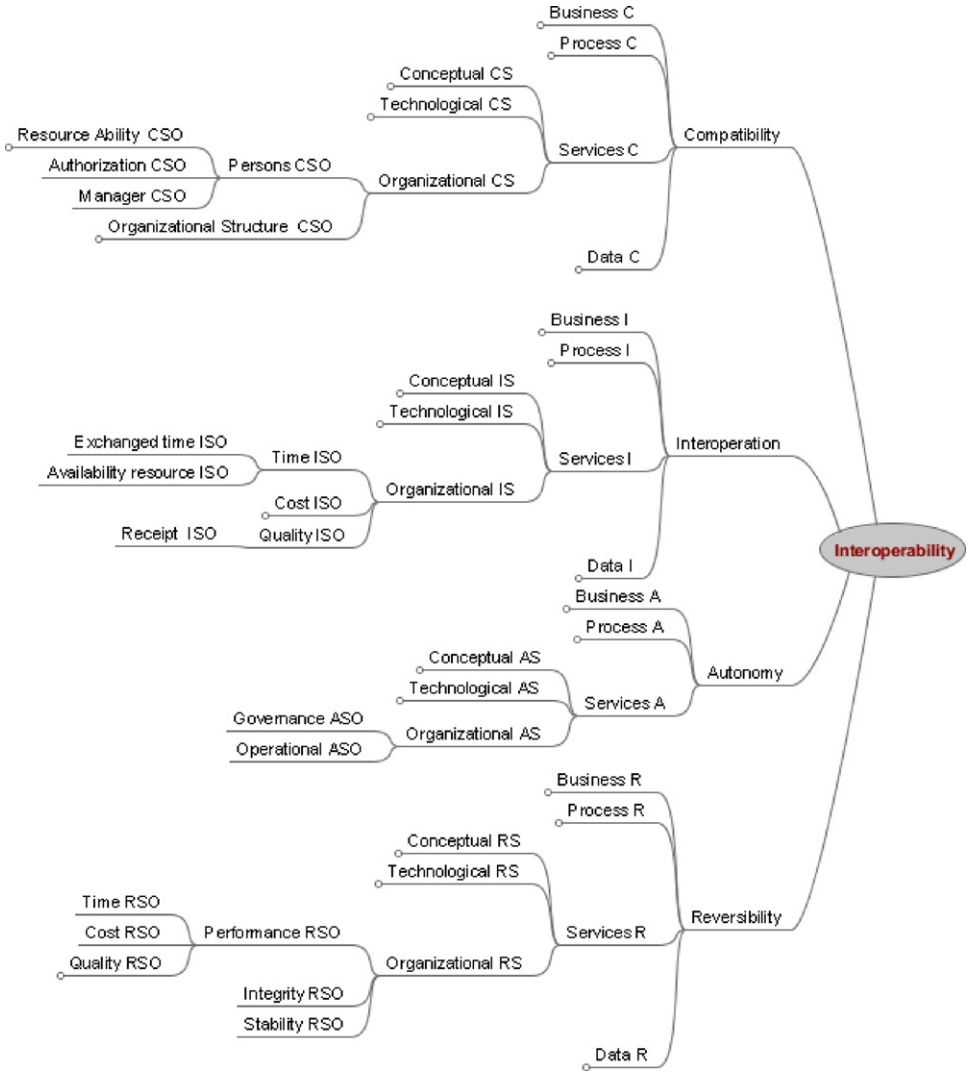


Fig. 4. Interoperability requirements represented on the GRADEI model (partial view).

performed using these few equations:

$$\text{Interoperability} = \text{Compatibility} \wedge \text{Interoperation} \wedge \text{Reversibility} \wedge \text{Autonomy} \quad (1)$$

$$\text{Compatibility} = \text{BusinessC} \wedge \text{ProcessC} \wedge \text{ServicesC} \wedge \text{DataC} \quad (2)$$

$$\text{ServicesC} = \text{ConceptualCS} \wedge \text{TechnologicalCS} \wedge \text{OrganizationalCS} \quad (3)$$

$$\text{ConceptualCS} = \text{SyntaxCDS} \vee \text{SemanticCDS} \quad (4)$$

In other words, by hypothesis, the overall interoperability of an enterprise requires each compatibility, interoperation, autonomy, and reversibility requirement to be respected as expressed in Eq. (1). In the same way, compatibility is respected if and only if each requirement related to the interoperability concerns (business, process, services, and data) are themselves respected as expressed in Eq. (2). Then, interoperability concerns are respected if and only if interoperability barriers are respected as expressed in Eq. (3) at the services level. Finally, conceptual requirements are respected if the syntax or the semantic requirement is respected as expressed in Eq. (4). However, if interoperability is required for only one concern (e.g., process interoperability), the user can select and check only equations related to this concern.

6. Verification of interoperability requirements

The objective of verification is to demonstrate that a set of selected interoperability requirements is satisfied. Indeed, the GRADEI model presented above allows users to select relevant requirements to be checked. In order to be able to perform this verification before the runtime of the collaborative process, this one is done on a model of the collaborative process. Several verification techniques exist in the literature such as informal verification techniques (test, simulation, and expertise) [20], and formal verification techniques (model checking and theorem proving) [21–23].

The simulation is done on a theoretical model whose behavior is considered to be similar to the behavior of the system concerned. It is done before the implementation of the system. However, simulation is unable to assume all of the system's behavioral scenarios. It requires human expertise to analyze results and formulate the demonstration. Nevertheless, simulation is now a well-known technique, which is increasingly developed and used in enterprises. The test is performed directly on an existing system. It can check, for example, capacity and relevance to detect errors before system implementation. Informal methods do not necessarily require the use of a model, but can be used directly on the real system to make the verification. These techniques can be used when a model of the system to analyze is not given or the

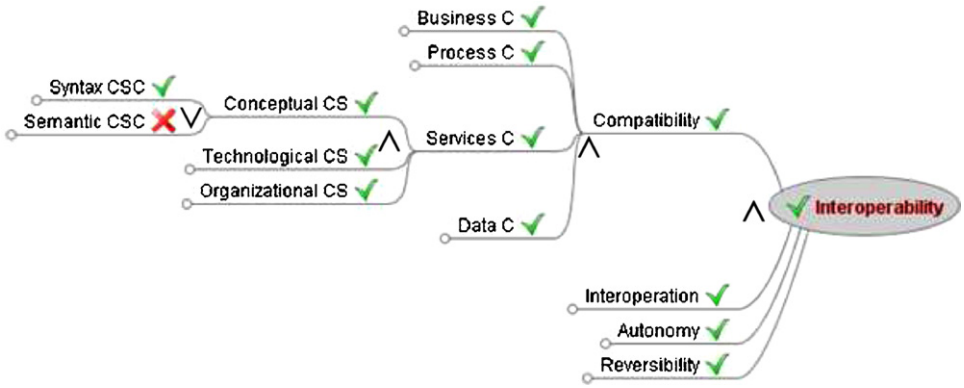


Fig. 5. Propagation of interoperability requirements analysis results using the GRADEI model.

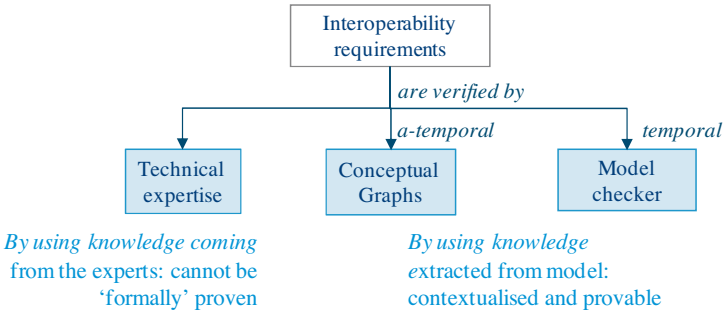


Fig. 6. Proposed verification techniques.

description of the requirement to formalize is complex. However, these methods require human skills and can be, obviously, subject to human error.

On the other hand, formal verification techniques can be used to explore a formal model exhaustively (i.e., a model obtained with a modeling language using a formal semantics). In this case, it is possible to provide a formal proof of whether or not a requirement is respected independently of any human interpretation. Formal verification techniques can be used to analyze the behavioral aspect of the collaboration, which offers a dynamic vision of the system. However, these methods can be difficult to implement (e.g., need for greater formalization) and are difficult to apply in some fields such as enterprise modeling (difficult to formalize models).

Other tools do exist, such as conceptual graphs, and can be used as verification techniques [24]. For instance, verification using conceptual graphs is based on mathematical mechanisms, such as projection, to ensure the veracity or simply the presence of knowledge in a given graph. However, conceptual graphs do not represent the behavior of a system, so they offer a static view of it.

Given the advantages and disadvantages of each method of verification, we have proposed to use two formal verification techniques in a complementary manner, and to also make use of technical expertise as summarized in Fig. 6.

The first verification technique is based on conceptual graphs [25] to verify a-temporal requirements. The advantage of using conceptual graphs is (1) to describe the collaborative process and interoperability requirements with the same language, (2) to have a convenient graphical form to handle, and (3) to have a mathematical foundation and mechanism (projection, rules, and constraints).

The second technique is based on model checking [23] for the temporal requirements. The advantage of using a model checker is (1) to include temporal aspects of the collaboration (i.e., behavioral model of collaborative process), (2) to consider all states in the collaborative process throughout the collaboration, and (3) to give formal proof of the problems that exist.

The final technique is based on expertise. This technique is deployed when interoperability requirements highlight particular points of view of the process, and cannot be described due to a limitation imposed by the modeling language. This aspect is not considered in this work.

Applying these techniques requires us to assume that the BPMN modeling language used to build the process model can be used to describe interoperability requirements. BPMN provides standardized notation that is readily understandable by all stakeholders involved in the design, development, and monitoring of a collaborative process. However, it is necessary to enrich this modeling language to embed the interoperability requirements model. The proposed conceptual enrichments described in [26] include interoperability concepts such as the nature of the flow exchanged (information, energy, material, or person), the availability of resources and their aptitudes. Furthermore, operational enrichments are made to study the behavior of all relevant BPMN elements.

The use of these verification techniques requires the collaborative process modeled with enriched BPMN to be translated – using model transformation rules – into an equivalent model upon which the formal verification techniques can be applied as shown in Fig. 7. Indeed, the proposed enriched version of BPMN suffers from a lack of formalization, and verification techniques cannot be applied directly with regards to interoperability. The first equivalent model was obtained using conceptual graphs. It enabled us to produce a-temporal requirements proof, which is presented in [26]. In this case, verification was performed with the COGITANT tool [27]. The second equivalent model was obtained

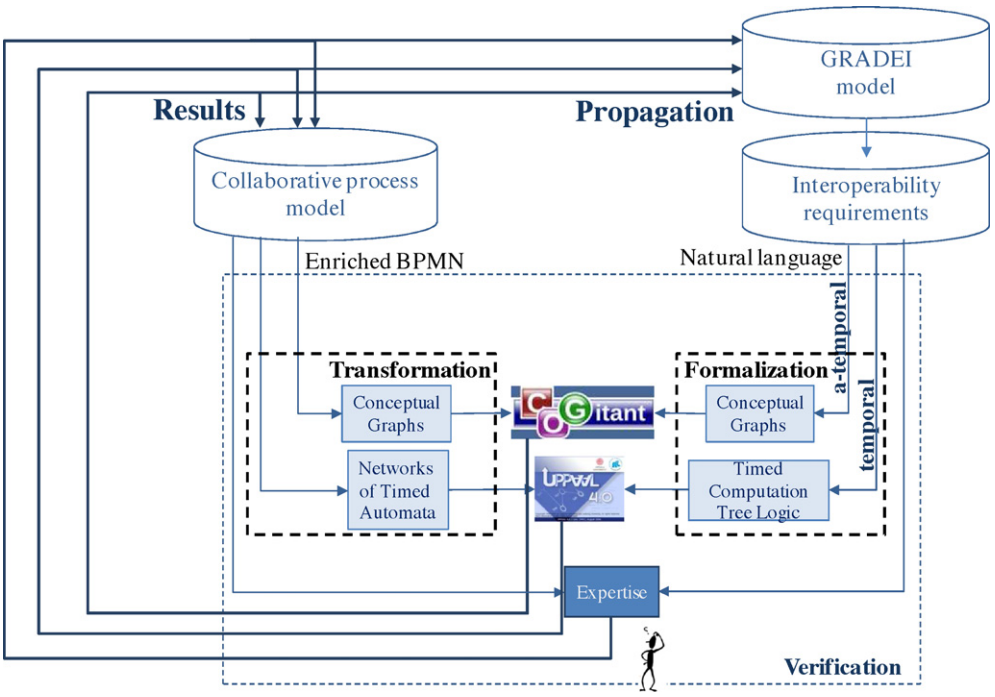


Fig. 7. Verification process for interoperability requirements.

Fig. 8. Example of a conceptual graph.

using a behavioral modeling language named Networks of Timed Automata for temporal requirements proof. In this case, the UPPAAL model checker is used for various reasons: the richness of TCTL temporal logic, it is an open source, user friendly, and stand alone tool [23]. In both cases of target models, the required rules for model transformation were developed with ATL (Atlas Transformation Language) [28] in order to re-write the collaborative process model into Conceptual Graphs and Networks of Timed Automata. In the case of the Conceptual Graphs, the objective is therefore to assume the coherence of the process model, that is to say to prove that each BPMN modeling entity used, and then instantiated in the process model, is well and completely defined. In the case of Networks of Timed Automata, the transformation rules have been established respecting an equivalence between BPMN entity behavior and state model entity behavior. Therefore, interoperability requirements were formalized to make their verification possible. Thus, a-temporal requirements are formalized with conceptual graphs and temporal requirements are formalized with TCTL. In addition, a non-formal verification technique was used to verify, by expertise, interoperability requirements that cannot be formally proven. Finally, the results of the three verification techniques are shown in the collaborative process model (modeled with enriched BPMN). These results were further exploited with the GRADEI model to verify abstract interoperability requirements using the principle of propagation.

6.1. Verification process for a-temporal requirements

A conceptual graph is defined as a graph with two kinds of nodes: concepts and oriented relations as shown in Fig. 8 with a conceptual graph that can be read as: "Any activities (concept) begin (relation) at a beginning date (concept)". Concepts and relations are described in hierarchical structures called concept and relation lattices. Individual markers are added to specify the model ("*" means generic concept).

The COGITANT tool can be used to handle conceptual graphs upon which the verification process of a-temporal requirements is based. The verification with conceptual graphs is based on the use of (1) a graph operation named projection, (2) a graph that represents a requirement (also named constraint graph) and, (3) a graph that describes the collaborative process model. The principle is to check to see if a constraint graph can be projected on the graph model of the process. If projection fails, the requirement is not verified and the causes can be highlighted by analyzing the resulting conceptual graph. To make verification with COGITANT, three types of files are necessary, which are called the support, fact, and constraint graphs.

The "*support*" graph represents all the concepts and relations from the enriched BPMN meta model and markers representing all the instances of these concepts and relations defined in the process

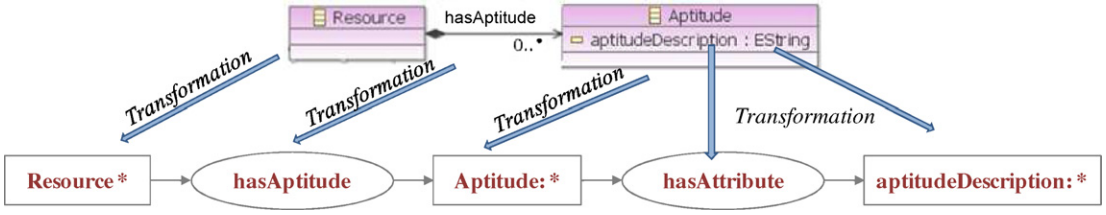


Fig. 9. Example of transformation from enriched BPMN to conceptual graph.

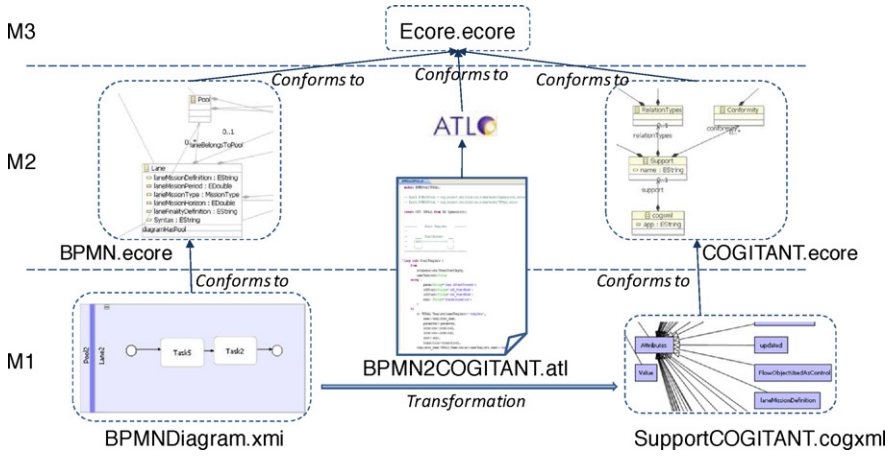


Fig. 10. Transformation from enriched version of BPMN to support in COGITANT.

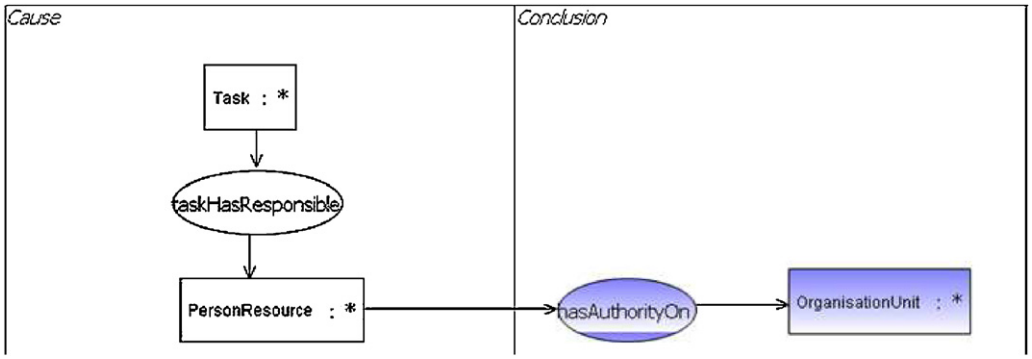


Fig. 11. Positive constraint representing a compatibility requirement.

model. The "*fact*" contains the equivalent conceptual graph of the model obtained by applying ATL transformation rules and respecting the support. Finally, the "*requirement*" to verify is modeled in another conceptual graph called the "*constraint*" graph. Verification is performed using the projection of a positive or a negative constraint on the conceptual graph that represents the model of the process studied. A positive constraint is described with a cause and a conclusion and its projection is performed according to the following interpretation: "*If the cause is true, then the conclusion must be true as well*". A negative constraint is a single conceptual graph and its projection is interpreted as: "*If a negative constraint is not projected on the fact model, it is verified*". The projection mechanism is the projection of a given requirement translated in the conceptual graph on the obtained conceptual graph that represents the translation of the model.

For the purpose of verification, we propose to transform each enriched BPMN element (UML class in the meta model) into a concept. Then, we propose to convert each attribute of each element (i.e., attributes of each class) as a concept also. Finally, relations between classes (in the enriched BPMN meta-model) are transformed into relations in the conceptual graph as shown in the Fig. 9.

Three ATL transformations must be performed to complete the transformation from process model to COGITANT. Hereafter, Fig. 10 represents the principle of the first transformation to get the support model (the two others transformations – fact and constraint – are based on the same principles and are not described here).

The first transformation procedure to obtain the support file (level M1) starts with the consideration of the meta models (level M2) of the enriched BPMN language and COGITANT, which conform, as well, to the e-core model (level M3). Thus, each class (including its attributes) is translated into a concept and each relationship in the meta-model is translated into a relationship in the support file. This transformation is made in order to provide all the needed concepts and relationships used and subsequently deployed in the fact file.

The second transformation is used to obtain a representation of the collaborative process model (fact) as a conceptual graph. Finally, the last transformation is performed to obtain constraints (representing the requirements) that have to be projected onto the equivalent graph model of the process. In this case, the requirement is translated into a positive or negative conceptual graph constraint depending on the user's intention.

For example, the compatibility requirement $CSO_{Manager}$ previously presented and described as: "*Each service has a clearly defined manager who has authority over the rest of the team*" can be formalized into a positive constraint as shown in Fig. 11.

The verification of this constraint using projection is performed with the projection of the cause onto the facts model. If the cause can be successfully projected, the conclusion must be projected too in order to respect the requirement.

6.2. Verification process for temporal requirements

The principle of a model checker is to verify properties exhaustively with temporized and possibly constrained automata that describe the behavior of a system. Obviously, here the system is the collaborative process model. Verification with model checkers requires two phases. The first phase consists in defining a set of equivalent behavioral models in the collaborative process model and in defining the collaborative process

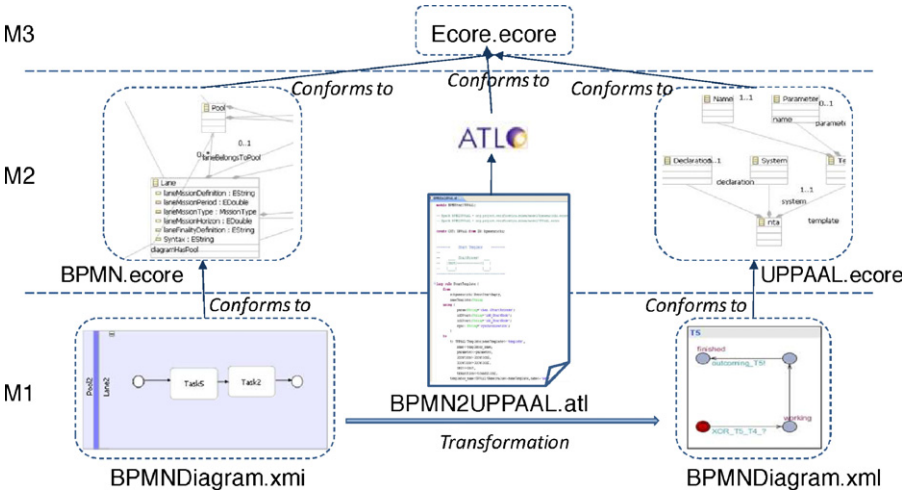


Fig. 12. Transformation from enriched version of BPMN to networks of timed automata.

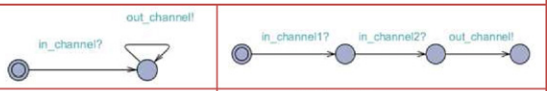
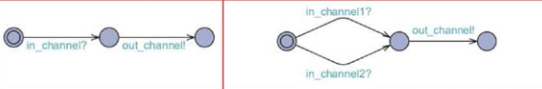
Enriched BPMN element	Existing NTA model	Templates
Start event	✓	
End event	✓	
Task	Native model	
AND (split/join)	✓	
XOR (split/join)	✓	
OR (split/join)	✗	-- --
Resource	✗	--
Several flows (in/out)	✗	--

Fig. 13. Existing model and model we developed.

model transformation rules to apply. The second phase consists in reformulating the temporal requirements under the form of properties respecting the formal language adopted by the chosen model checker (in this study, a temporal logic) [29].

The UPPAAL tool can then be used to handle a behavioral model defined as a set of templates, which communicates with synchronization (either in the form *Expression*! for sending or *Expression*? for receiving synchronization), using channels and syntax like sent/receive. Each template has locations and transitions to link a location source to a target source [23].

The enriched BPMN model must be transformed into Networks of Timed Automata to perform verification of temporal requirements. In Fig. 12, the model transformation procedure (level M1) starts with the consideration of the meta models (level M2) of the enriched BPMN language and UPPAAL which conform, as well, to the e-e-e-core model (level M3).

This transformation is made in order to provide all the necessary concepts that are used and deployed in the Networks of Timed Automata. In this way, it is mandatory to consider all the modeling entities which will be used in the checking task. Thus, each class (including its attributes) in the meta-model is translated into templates. Respecting this consideration, each BPMN element can be extracted from the collaborative process model in order to produce the corresponding template representing Networks of

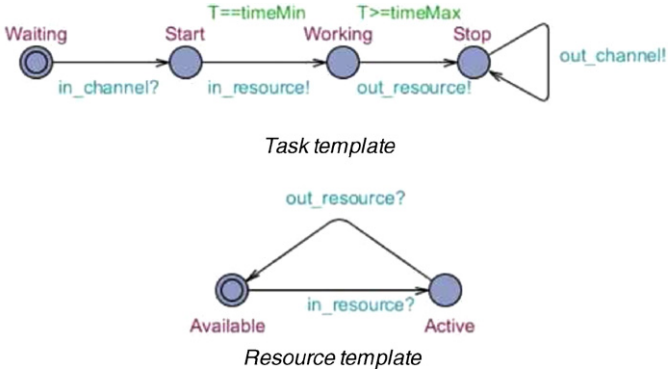


Fig. 14. Task template and resource template.

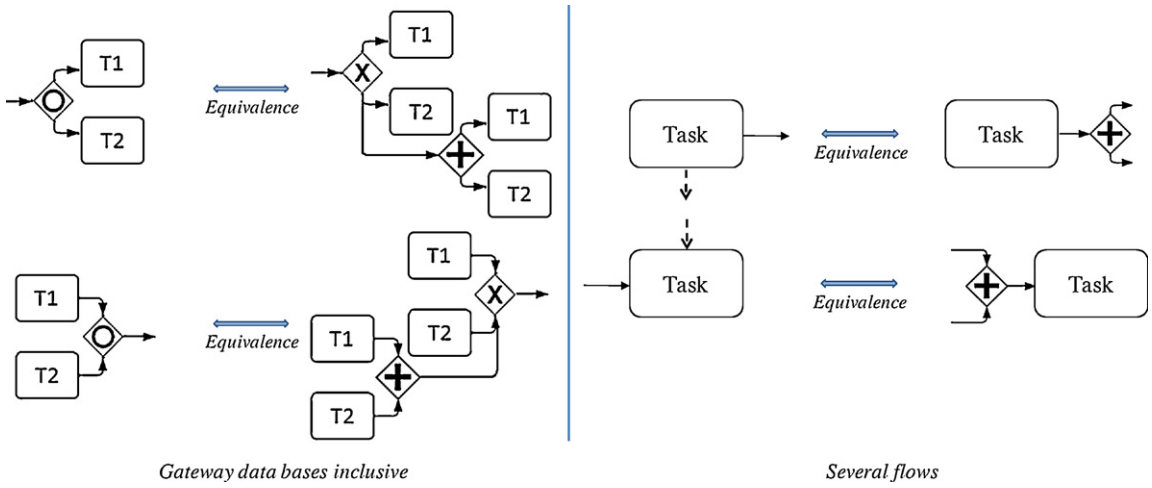


Fig. 15. Behavior of a gateway data based inclusive and many (in/out) flows.

Timed Automata. Thus, these templates gather all the knowledge described in the model, and represent the collaborative process behavioral model.

Some behavioral models of BPMN elements, which use Networks of Timed Automata, can be found in the literature. For instance, [30] propose the behavioral model of the BPMN elements such as: start and end event, "*gateway data based parallel*" (AND), the "*gateway data based exclusive*" (XOR), and the native model of Task. For instance, the behavior of a task is represented by four locations and two synchronizations as presented in Fig. 13. In this case, transformation rules are defined directly. Other behavioral models that do not exist, but that are considered to be fundamental in collaborative processes, are being developed. Likewise, in this research work, we develop other behavioral models corresponding to (1) the task when a resource is involved, (2) the resource, (3) the "*gateway data based inclusive*" (OR) and (4) consideration of many in/out flow (including sequence and/or message flows) on a task. Then, for each enriched BPMN element, a transformation using ATL is defined and performed to obtain the behavioral model in the form of Networks of Timed Automata.

For instance, the translation of a task when it uses a resource and the translation of a resource are presented Fig. 14. Initially, the

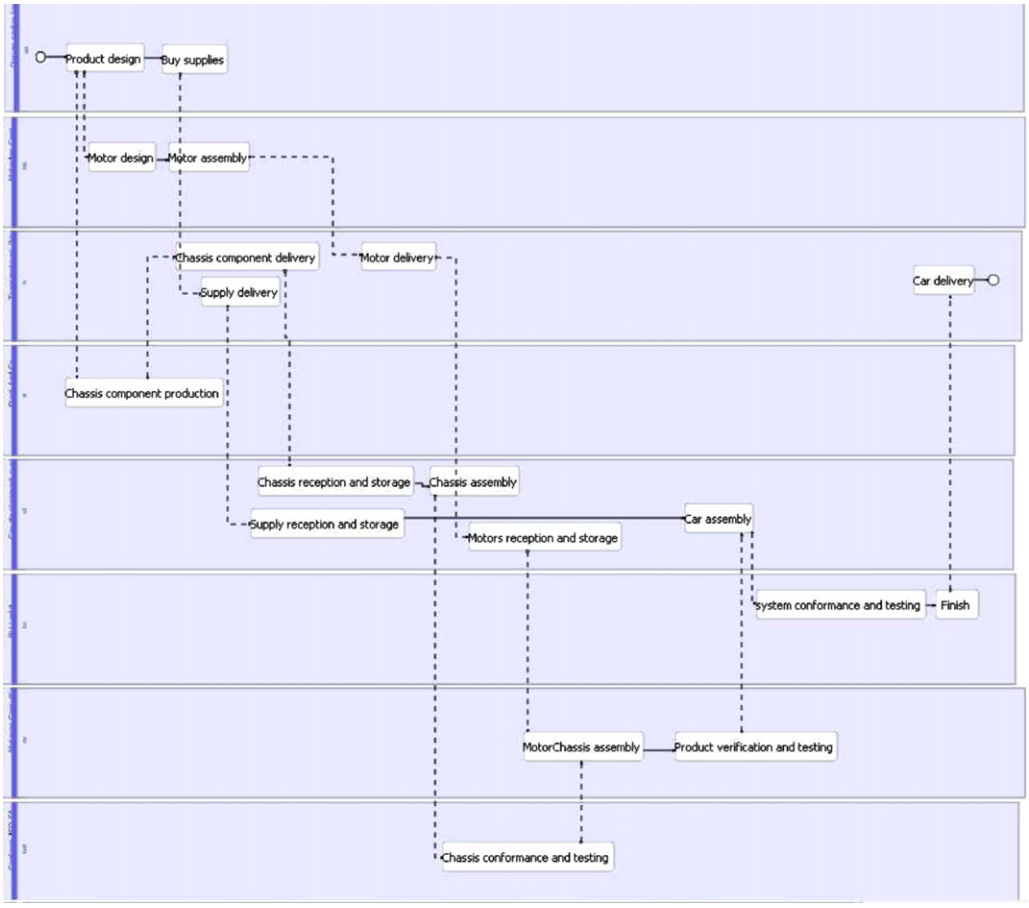


Fig. 16. Model of the collaborative process with enriched BPMN (simplified view).

task is waiting to start. Then after starting, it uses the resource which is available at the beginning by sending it a synchronization such as "in_resource!". Subsequently, the resource returns to its initial state by receiving a synchronization "out_resource?" when the task has finished using it. Finally, the task stops.

Furthermore, based on the templates given by [30] for the *gateway data based parallel* (AND) and the "*gateway data based exclusive*" (XOR), the behavior of the "*gateway data based inclusive*" (OR) and the consideration of many (in/out) flows is proposed as shown in Fig. 15. As a consequence, we propose to represent the behavior of the "*gateway data based inclusive*" by the use of a "*gateway data based exclusive*" and "*gateway data based parallel*". Then, we propose to simulate the existence of a "*gateway data based parallel*" to represent the in/out of many flows (sequence flow and/or message flow) of a task.

To enable the implementation of formal verification techniques, the temporal requirements are formalized into TCTL properties (Timed Computation Tree Logic, i.e., the UPPAAL property specification language) [23]. TCTL is an extension of CTL (Computational Tree Logic) which can be used to consider several possible futures based on the state of a system. The UPPAAL model checker has four TCTL quantifiers (A: for all paths, E: a path exists, [ ]: all states in a path, $< >$: some states in a path), which can be used to write a property p:

- $E< > p$: reachability (i.e., it is possible to reach a state in which p is satisfied).
- $A[\ ]\ p$: invariantly p (i.e., p is true in all reachable states).
- $A< > p$: inevitable p (i.e., p will inevitably become true).
- $E[\ ]\ p$: potentially Always p (i.e., p is potentially always true).
- $P \rightarrow q$: p leads to q (i.e., if p becomes true, q will inevitably become true).

According to the UPPAAL property specification language defined above, the temporal requirements written in natural language are manually re-written into properties using TCTL. Then

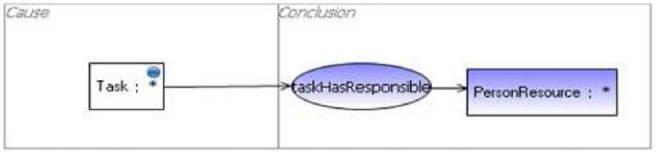
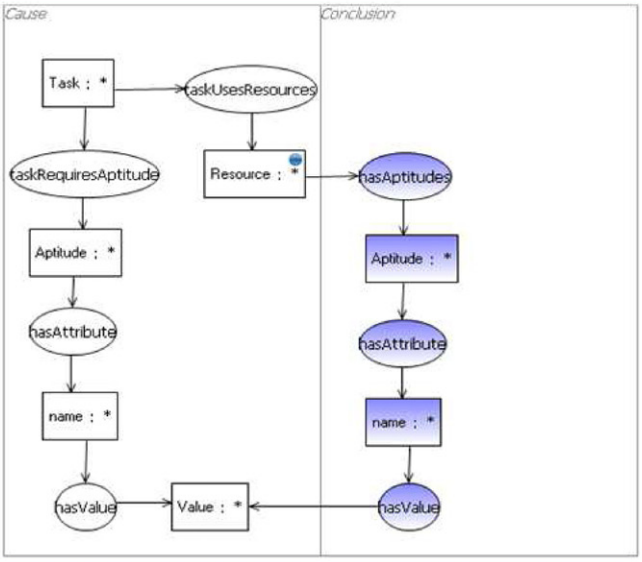
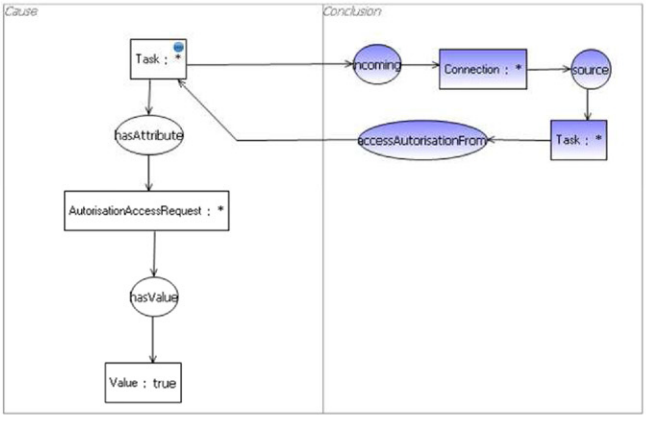
Compatibility requirements	Conceptual Graphs
$CSO_{Manager}$	
$CSO_{Aptitude}$	
$CSO_{Authorization}$	

Fig. 17. Formalization of compatibility requirements.

the UPPAAL model checker exhaustively verifies properties in TCTL through all the execution paths of the behavioral models that are reachable.

For instance, the $ISO_{ExecutionTime}$ requirement described as: "*Resource is active throughout the duration of execution of the activity* ($5 < T < 10$)" can be formalized into a property using TCTL as:

"$E<>$ *Resource.Active and Task.Working and* $T > 5$ *and* $T < 10$"

This property indicates that a path can exist, where the resource is active and the task is in the state Working between $5 < T < 10$. This property can be verified with the templates represented Fig. 14. The next part presents an application case study.

7. Application case study

The collaborative process we propose is shown in Fig. 16. It aims to design and produce vehicles. Various geographically distributed partners in the European territory wish to anticipate potential defects inherent in their interoperability.

The compatibility requirements to be verified are related to the organizational barrier at the level of services and are expressed as follow:

- $CSO_{Manager}$: "*Each service has a clearly identified* manager."
- $CSO_{Ability}$: "A *resource has the ability to do activity requested*."
- $CSO_{Authorization}$: "*The sender has the necessary authorizations to interact with the receiver*."

These compatibility requirements are a-temporal. They must be verified with conceptual graphs. As a result, they must be formalized into conceptual graphs as shown in Fig. 17.

The compatibility requirement $CSO_{Manager}$ is formalized using a positive constraint. This constraint expresses the fact that each service has a manager. However, it does not indicate if this manager has been identified. Therefore, this requirement can be decomposed into three compatibility requirements described as follow:

- $CSO_{ManagerName}$: "The manager of a *service has a name*."
- $CSO_{ManagerPhone}$: "The manager of *a service has a phone number*."
- $CSO_{Manager}$: "The manager of *a service has an*"

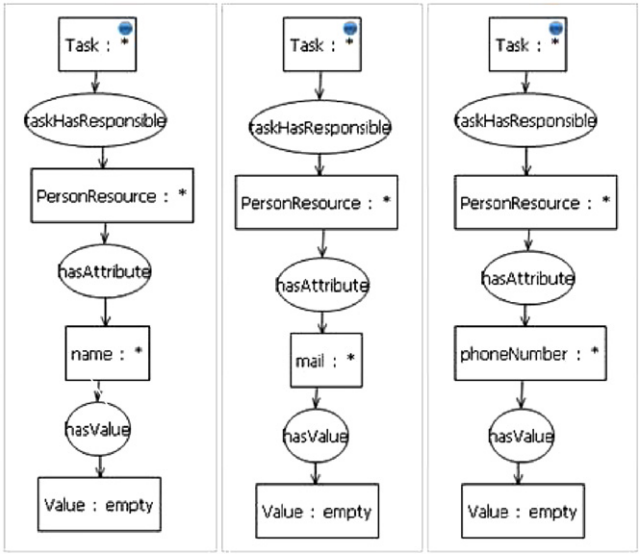


Fig. 18. Formalization of $CSO_{Manaager}$ with three negative constraints.

These compatibility requirements give more information about the manager's name, phone number, and they can be formalized with three negative constraints as shown in Fig. 18.

The compatibility requirement $CSO_{Aptitude}$ is formalized using a positive constraint. This constraint is expressed as follows: for each service (task), the resources used by this service must have the required skills.

The compatibility requirement $CSO_{Authorization}$ is formalized using a positive constraint. This constraint reflects the fact that each task interacting with another task requires the resources involved to have the relevant authorizations to perform the exchanges.

The interoperation requirements to verify are related to the organizational barrier at the services level, and are expressed as follow:

- $ISO_{TimeExecution}$: "*Resource is active throughout the duration of execution of the activity*."

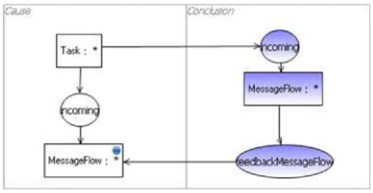
Interoperation requirements	Formalization
$ISO_{TimeExecution}$	Motor Assembly : "*E<> Motorsassembling_mic_MotorIncCorp.Working and PersonforAssembling.Active and T<Motorsassembling_mic_MotorIncCorp.timeMax and T>Motorsassembling_mic_MotorIncCorp.timeMin*" Chassis Assembly : "*E<> Chassisassembling_cd_CardevelopmentsystemsLmtd.WorkingandPersonforAssembling.ActiveandT<Chassisassembling_cd_CardevelopmentsystemsL mtd.timeMax and T>Chassisassembling_cd_CardevelopmentsystemsLmtd.timeMin* "
$ISO_{TimeExchange}$	"*Start_Productdesign.Start → forall (i: NbTask) Reception [i] - Emission [i] <6*"
$ISO_{Receipt}$	

Fig. 19. Formalization of interoperation requirements.

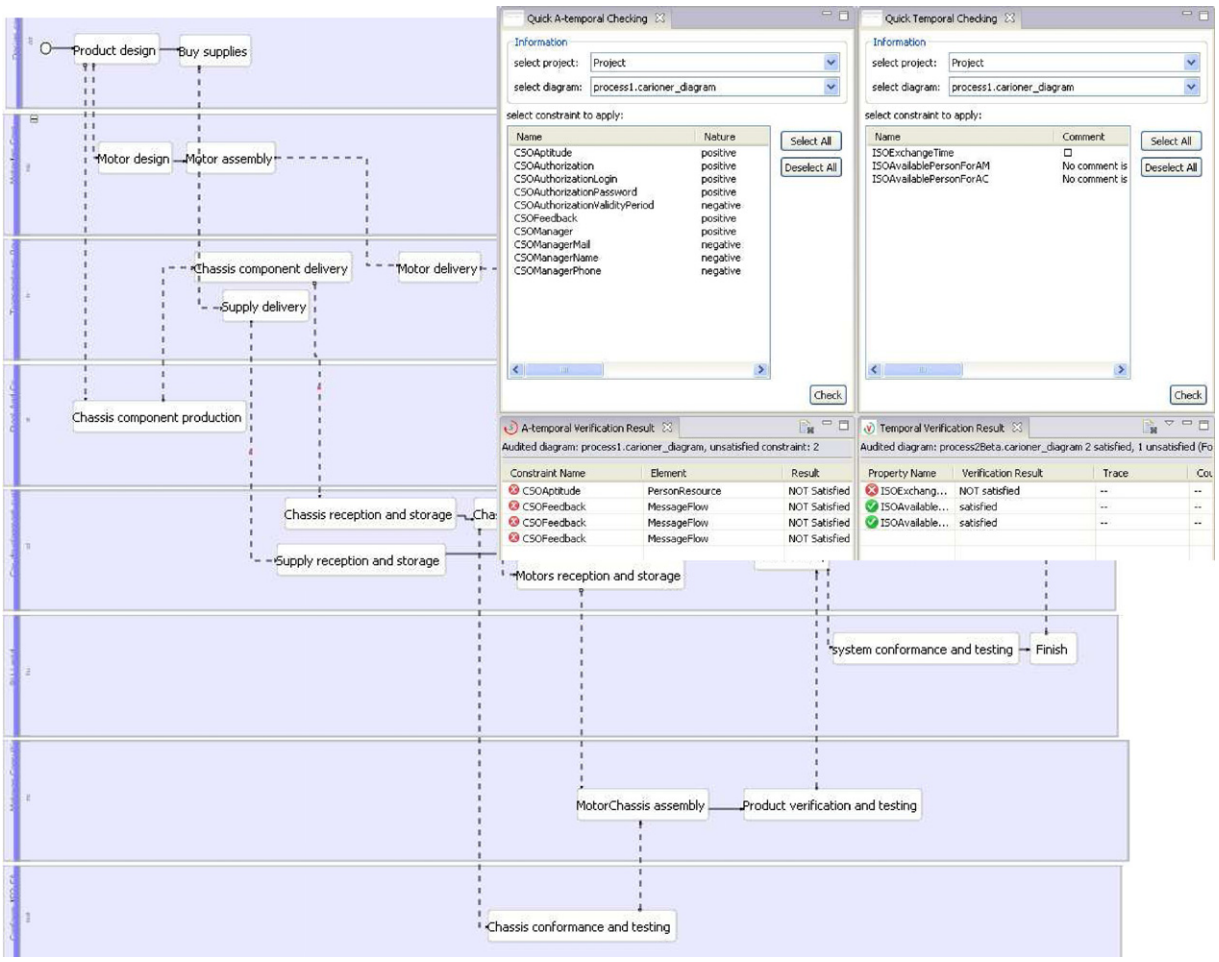


Fig. 20. Results of the interoperability requirements verification.

- $ISO_{ExchangeTime}$: "The time *for exchange between services is strictly less than n time units (TU).*"
- $ISO_{Receipt}$: "The receiver *sends a receipt to the sender.*"

The first and second interoperation requirements have temporal aspects and must be formalized with TCTL to make their verification possible with the UPPAAL model checker. The final interoperation requirement has an a-temporal aspect and must be formalized with conceptual graphs to verify it using the COGITANT tool. The first interoperation requirement $ISO_{TimeExecution}$ is used to verify if the activities "*motor assembly*" and "*chassis assembly*" can use the same resource. As a consequence, this requirement has temporal aspects and is formalized using the two properties shown in Fig. 19.

The second requirement $ISO_{ExchangeTime}$ is chosen to be verified to inform interacting enterprises if the exchange time between their activities is respected and less than the desired exchange time limit. This temporal requirement is formalized by a property using TCTL. The final interoperation requirement $ISO_{Receipt}$ is used to ensure that the receiver has received stocks. This requirement has an a-temporal aspect and is formalized by a constraint with a Conceptual Graph.

Furthermore, we propose to verify an autonomy requirement and a reversibility requirement. These requirements are expressed at the service level for the organizational barrier and are expressed as follows:

- $ASO_{Resource}$: "A service can *replace a resource used by another service in the public collaborative process.*"
- $RSO_{PerformanceResource}$: "The *resources used in the collaborative process have the same performance level – including tolerances – as before the collaboration.*"

$ASO_{Resource}$ expresses the fact that a service is able to replace its resource if it is used by another one. $RSO_{PerformanceResource}$ represents the ability of a resource to retrieve its past performances. These requirements cannot be verified using formal verification techniques; however, they can be verified using the expertise technique. Finally, the result of the verification using the UPPAAL and COGITANT tool for the verification of compatibility and interoperation requirements is shown Fig. 20.

The results of a-temporal requirements verification are shown on the left side of Fig. 20. The requirements $CSO_{ManagerName}$, $CSO_{ManagerPhone}$ and $CSO_{Manager}$are satisfied by all tasks (no errors are reported). This means that the manager is known and identified. The $CSO_{Aptitude}$ requirement was not satisfied by all tasks. Indeed, we can note that the "*Motor assembly*" and the "*Chassis assembly*" tasks use the same resource. The first task requires the "*Motor component assembly*" ability while the second task requires the "*Component assembly*" ability. However, the resource only has the "*Motor component assembly*" ability. As a result, the task should change the resource. Through the detection of this problem using the approach, the change of the resource can be made before the execution of the process. If this change is not made before running the collaborative process, this problem can hinder the collaborative process in terms of execution time to make this change after it has been detected. The verification of the $CSO_{Authorization}$ requirement indicates that it has been satisfied by all activities that require authorizations.

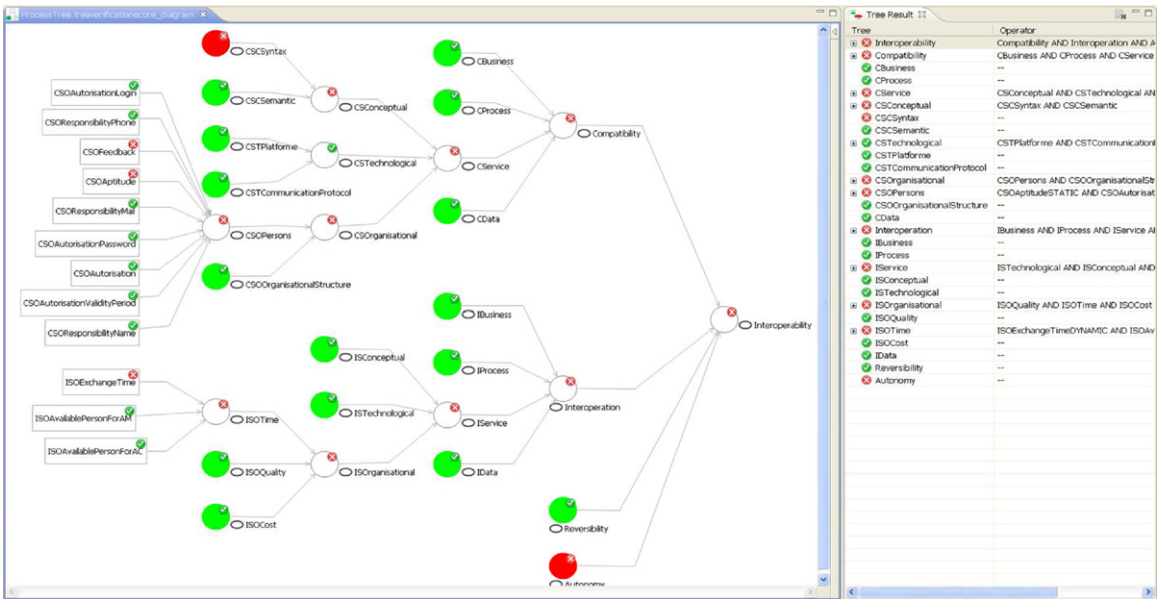


Fig. 21. Propagation using the GRADEI model.

Finally, verification of the $ISO_{Receipt}$ interoperation requirement indicates that the "*Chassis reception and storage*", "*Supply reception and storage*", *and* "*Motor reception and storage*" tasks do not return a receipt for the tasks that provided them supplies. The partner that is in charge of delivering the components for the other partners must ensure that the delivery of each component is completed successfully to avoid possible costs for re-delivery and related delays.

The verification of temporal requirements is shown on the right of Fig. 20. The two properties representing the $ISO_{TimeExecution}$ requirement for the "*Motor assembly*" and "*Chassis assembly*" tasks are satisfied. Indeed, one resource is used by the two tasks at different periods. However, it is important to keep in mind that this resource does not have the skills required by one task. As a result, the ability of the two tasks to use the same resource does not mean that it can be used. In conclusion, it is important to take all the interoperability requirements into consideration because the verifications with both tools are complementary. Finally, the verification of the $ISO_{ExchangeTime}$ requirement can be used to conclude quickly on the exchange time between all tasks. The result indicates that one of tasks does not respect the mandatory exchange time. The verification of these interoperability requirements provides local detection, which can be used subsequently to solve the problems detected. However, this local detection is not sufficient. Indeed, these problems can impact the collaboration at a global level. As a result, we propose to use propagation mechanisms with the GRADEI model as shown in Fig. 21.

The results of the verification with the UPPAAL and COGITANT tools for the temporal and a-temporal requirements are used by the GRADEI model (requirements represented on the far left). Requirements modeled in green (satisfied) and red (not satisfied) represent the result of the expertise. The expert can validate or not validate a requirement directly in the GRADEI model. The logical function that links the abstract requirements to more specific requirements is the logical 'AND' operator. The result of the propagation indicates that even if a resource is available to perform a task, if it does not have the required skills, the collaborative process has interoperability problems. In addition, the non-verification of these requirements means that the more abstract requirement "interoperability" is not satisfied.

8. Conclusion

In a collaborative context, interoperability is becoming increasingly important. A partner involved in collaboration with other partners aims to detect and solve interoperability problems as quickly as possible. The approach we propose aims to formulate, formalize, and check interoperability requirements in order to detect locally and globally interoperability problems in an anticipative way that uses verification techniques. The formulation of interoperability leads to define four categories related to the compatibility, interoperation autonomy and reversibility. These categories are then split up and structured using the GRADEI model. The analysis of interoperability requirements in a collaborative process model uses model transformation techniques, formal verification techniques for local verification and propagation mechanism for global verification. Local verification is performed using complementary verification techniques. The first verification technique uses conceptual graphs to verify a-temporal interoperability requirements. The second verification technique is a formal verification technique using model checking to verify temporal interoperability requirements. Finally, a non formal verification technique such as expertise is applied to verify requirements that cannot be formally verified. The local detection of interoperability requirements is insufficient. Indeed, the result of the verification of a requirement can influence the interpretation of the result of another requirement. So the global verification is based on the propagation of the results of the local verification by interpreting some parts of the GRADEI model. Future research must first analyze the verification of reversibility and autonomy requirements. This verification can be done with a complementary simulation approach based on distributed multi-agent systems [31] to improve interoperability problem detection. Finally, our research work aims to find solutions for the interoperability problems detected.

References

[1] J. Touzi, «Aide à la conception de Système d'Information Collaboratif: support de l'interopérabilité des entreprises», PhD Thesis, Institut National Polytechnique de Toulouse, November 2007 (in French).
[2] B. Aubert, A. Dussart, Système d'Information Inter-Organizationnel, Rapport Bourgogne, Groupe CIRANO, March 2002 (in French).

[3] A. Esper, «Intégration des approches SOA et orientée objet pour modéliser une orchestration cohérente de services», PhD Thesis, Institut National des Sciences Appliqués de Lyon, September 2010 (in French).
[4] ISO/DIS 11345-1, Advanced automation technologies and their applications, Part 1: Framework for enterprise interoperability, 2009.
[5] IDEAS Project Deliverables, «*WP1-WP7*», Public reports, 2003.
[6] INTEROP: Enterprise Interoperability-Framework and knowledge corpus – Final report, INTEROP NoE, FP6 – Contract no. 508011, Deliverable DI.3, May 21st 2007.
[7] ATHENA Integrated Project, «Guidelines and best practices for applying the ATHENA interoperability framework to support SME participation in digital ecosystems», ATHENA deliverable A8.2, 2007.
[8] C4ISR Architecture Working Group Levels of Information Systems Interoperability (LISI), United States of America Department of Defense, Washington DC, USA, 30 March, 1998.
[9] A. Tolk, J.A. Muguira, The Levels of Conceptual Interoperability Model, Proceedings of Fall Simulation Interoperability Workshop (SIW), Orlando, USA, 2003.
[10] T. Clark, R. Jones, Organizational Interoperability Maturity Model for C2, Proc. of Command and Control Research & Techn. Symposium, Newport, USA, 1999.
[11] N. Daclin, D. Chen, B. Vallespir, Methodology for enterprise interoperability, in: 17th IFAC World Congress (IFAC'08), Seoul, Korea, 2008.
[12] C.T. Ford, «Interoperability measurement», PhD Thesis, Department of the Air Force Air University, Air Force Institute of Technology, 2008.
[14] BPMN, Business Process Modeling Notation, V1.2, http://www.bpmn.org/, 2009.
[15] NEHTA, «Interoperability maturity model», Version 1.0, 26 March 2007.
[16] INCOSE, «System Engineering (SE) Handbook Working Group, System Engineering Handbook, A «How To»», Version 3.1, Guide For All Engineers, http://www.incose.org, 2007.
[17] ISO/IEC 15288:2008(E), «IEEE Standards 15288.2008 – Systems engineering – System life cycle processes (2nd edition)», 2008.
[18] S.J. Scucanec, J.R. Van Gaasbeek, A day in the life of a verification requirement, in: U.S. Air Force T&E Days, Los Angeles, California, February, 2008, 2008.
[19] M. Kasunic, W. Anderson, Measuring systems interoperability: challenges and opportunities, Software engineering measurement and analysis initiative, Technical note CMU/SUE-2004-TN-003, 2004.
[20] O. Balci, W. Ornwsby, Expanding our horizons in verification, validation and accreditation research and practice, in: E. Yücesan, C.-H. Chen, J.L. Snowdon, J.M. Charnes (Eds.), 2002 Winter Simulation Conference, 2002.
[21] M. Edmund, Clarke Jr., O. Grumbereg, A.P. Doron, Model Checking, The MIT Press, 1999.
[22] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph Schnoebelen, P. McKenzie, Systems and Software verification: Model Checking Techniques and Tools, Springer, 2001.
[23] G. Behrmann, A. David, K.G. Larsen, A Tutorial on Uppaal, Department of Computer Science, Aalborg University, Denmark, 2004.
[24] V. Chapurlat, B. Kamsu-Foguem, F. Prunet, «Enterprise model verification and validation: an approach», Annual Review in Control, vol. 27, no. 2, pp. 185–197, 2003.
[25] J.F. Sowa, Conceptual graphs, IBM Journal of Research and Development (1976).
[26] M. Roque, V. Chapurlat, Interoperability in collaborative processes: requirements characterisation and proof approach, in: PRO-VE'09, 10th IFIP Working Conference on VIRTUAL ENTERPRISES, Thessaloniki, Greece, 7–9 October, 2009, 2009.
[27] Cogitant CoGITaNT Version 5.2.0, Reference Manual, http://cogitant.sourceforge.net, 2009.
[28] ATLAS Groupe INA & INRIA Nantes ATL Atlas Transformation Language, Specification of the ATL Virtual Machine, Version 0.1, 2005.
[29] R. Alur, C. Courcoubetis, D. Dill, Model-checking in dense real-time, Information and Computation 104 (1) (1993) 2–34.
[30] V. Gruhn, R. Laue, Using timed model checking for verifying workflows, Computer Supported Activity Coordination 7 (2005) 5–88.
[31] A.S. Rebai, V. Chapurlat, System interoperability analysis by mixing system modeling and MAS: an approach, agent-based, technologies and applications for enterprise interoperability (ATOP), in: Eighth International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2009), Budapest, Hungary, May, 2009, 2009.