HAL
open science

Formalisation and verification of interoperation requirements on collaborative processes

Sihem Mallek, Nicolas Daclin, Vincent Chapurlat

To cite this version:

Sihem Mallek, Nicolas Daclin, Vincent Chapurlat. Formalisation and verification of interoperation requirements on collaborative processes. 18th IFAC World Congress, Aug 2011, Milano, Italy. 10.3182/20110828-6-IT-1002.01796 . hal-00588832

HAL Id: hal-00588832

https://hal.science/hal-00588832v1

Submitted on 20 Jun 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Formalisation and verification of interoperation requirements on collaborative processes

Sihem Mallek*, Nicolas Daclin*, Vincent Chapurlat*

** Laboratoire de Génie Informatique et d'Ingénierie de Production - LGI2P - site de l'Ecole des Mines d'Alès, Parc Scientifique Georges Besse, F30035 Nîmes Cedex 5, France (Tel (+33) 466 387 066 - e-mail: surname.name@mines-ales.fr)*

Abstract: Interoperability is became a crucial question to improve success of a collaboration in a networked enterprise. Therefore, in a collaborative context, enterprises have to detect their interoperability problems to solve and to reach an efficient collaboration. This research work aims then to define, to formalise and to analyse a set of interoperability requirements that each partner of a collaborative process have to satisfy prior to any collaboration. This paper focuses and illustrates how interoperation requirements related to the dynamic aspect of the collaboration may be formalised and verified by the use of a formal verification technique.

Keywords: interoperability, interoperability requirements, collaborative process, verification technique, model checker.

1. INTRODUCTION

In the current globalized and aggressive market environment, enterprises are more and more involved in collaborative processes. A collaborative process can be defined as "*a process whose activities belong to different organisations*" (Aubert *et al.*, 2002). The efficiency of this kind of process leads enterprise to assume its interoperability in particular prior to the collaboration. Therefore the goal is to minimise possible defaults and risks induced by a lack of interoperability, and to evaluate the potential performance of the collaboration. For this (Daclin *et al.*, 2008) proposes to define and assess performance criteria such as: the cost, the quality and the time of the interoperation.

The research presented in this communication aims to help managers and engineers in networked enterprise to detect possible interoperability problems. Interoperability requirements are proposed and an engineering approach allowing to establish an interoperability requirements reference repository is developed. Moreover, some conceptual extensions to existing processes modelling languages to consider the notion of interoperability requirement are proposed (Roque *et al.*, 2009). This enrichment is necessary to perform the verification of interoperability requirements on the resulting collaborative process model. A verification approach based on a formal checking approach is then highlighted.

This paper focuses on the formalisation and verification of "*interoperation requirements*" that characterize the expectations to be verified taking dynamic aspect of the collaboration into account. It is structured as follows. Section 2 reminds the principles and classification of interoperability requirements. Section 3 introduces the proposed mechanisms used to analyse interoperability requirements. Section 4 presents the generation of the behavioural model to verify interoperation requirements. The formalisation of interoperation requirements is given section 5. To illustrate the verification of interoperation requirements, an application case is given in section 6 before presenting some outlines perspectives of this research work.

2. INTEROPERABILITY REQUIREMENTS REFERENCE REPOSITORY

Several definitions of enterprise interoperability are proposed in the literature. (ISO/DIS 11354-1, 2009) defines enterprise interoperability as the "*ability of enterprises and entities within those enterprises to communicate and interact effectively*". In this way, interoperability is seen as the ability for a system (here an organizational unit such as team, enterprise, collaborative process), to work efficiently in collaboration with any others systems. Thus, interoperability can be considered as a set of requirements to satisfy prior any collaboration. 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*" (Scucanec *et al.*, 2008).

As far as the two dimensions (*i.e.* interoperability barriers and interoperability concerns) of the interoperability framework developed in the Network of Excellence INTEROP (INTEROP, 2007) are considered, three classes of interoperability requirements are defined (Mallek *et al.* 2010): *compatibility*, *interoperation* and *reversibility* requirements.

In fact, interoperability remains often related to *compatibility requirements*. Compatibility means to harmonize enterprises (method, organization, tool...) together. For instance, heterogeneous information exchanged can be understood and

exploited by each ones without interfacing effort. However, the compatibility represents only the static aspect of the collaboration that may be checked definitely and independently from the dynamic of the entities involved in the collaboration. The dynamic aspect of the collaboration is thus described by the "*interoperation requirements*". The Interoperation focuses on the abilities of the (part of the) enterprise to adapt its organisation, its operation modes and its behaviour when it interacts. In other words, it concerns the runtime phase of the collaborative process. Furthermore, when collaboration takes over, partners wish to retrieve their autonomy while remaining efficient. Indeed, any collaboration can induce a modification of the organisation or of the behaviour of one of the entities in order to collaborate. As a consequence, it is necessary to describe another kind of requirements called "*reversibility requirements*". Reversibility means that an enterprise may maintain or retrieve easily its autonomy at the end of any collaboration. In summary, interoperability requirements can be classified into three classes that are, for example, in agreement with the creation, operation and dissolution phases of a virtual enterprise (Camarinha-Matos *et al.*, 2003). Indeed, when a business opportunity is detected during the creation phase, compatibility is required. Then, an efficient interoperation is necessary through the operation phase. Finally, reversibility makes its all sense on the dissolution phase where enterprises aim to retrieve their own autonomy in order to carry on their own operations or to go to another collaboration. These classes are defined as follow:

- A compatibility requirement is defined as "*a statement that specifies a function, ability or a characteristic, independent of time and related to interoperability barriers (conceptual, organizational and technological) for each interoperability concerns (data, services, processes and business), that enterprise must satisfy before collaboration effectiveness*".
- An interoperation requirement is defined as "*a statement that specifies a function, ability or a characteristic, dependent of time and related to the performance of the interaction, that enterprise must satisfy during the collaboration*".
- A reversibility requirement is defined as "*a statement that specify functions, abilities or characteristics related to the capacity of enterprise to retrieve its autonomy and to back to its original state (in terms of its own performance) after collaboration, that enterprise must satisfy*".

The description and the handle of the requirements expressed in each category remains a difficult task. Therefore, the proposed approach allows to dispose of a requirement model for describing the expected abilities without ambiguity, thanks to an interoperability requirements reference repository. This repository enables to help the user for structuring and organising its own interoperability requirements and to reuse existing requirements. This repository is described through a causal tree model. It is an oriented graph G formalised as follow:

G ::= (L, N, N_0) With:

- L = {L_j / j ∈ [0,n] ; L_j ::= (*SourceN*, *TargetN*) with (*SourceN*, *TargetN*) ∈ N x N, *SourceN* ≠ *TargetN* and level(SourceN) > level(TargetN)}
- N = {N_i / i ∈ [0,m] ; N_i ::= ($name_i$, $description_i$, $relation_i$, $fact_i$, $level_i$, $value_i$)} where :
  - ($name_i$, $description_i$) ∈ String x String
  - $relation_i$ ::= (*quantifier*, *T*, θc, θe) with :
    - *quantifier* ∈ {"∀", "∃"}
    - $T \subseteq \boldsymbol{T}$ / set of possible moments
    - $\theta e : fact_i \cup T \rightarrow \{0 , 1\}$

      $\theta e : \{f_1, f_2, ...\} \cup \{t_1, t_2, ...\} \rightarrow \{0 , 1\}$
    - $\theta c : \mathbf{N}_{Ni} \rightarrow \{0 , 1\}$ with $\mathbf{N}_{Ni}$ ::= {N_j ∈ N / j ∈ [0,m], j ≠ i, ∃ L_k (N_i, N_j) / k ∈ [0,n] } is the set of Source nodes of N_i.

      $\theta c : \{value(\mathbf{N}_{N1}), value(\mathbf{N}_{N2}), ...\} \rightarrow \{0 , 1\}$

In other words, a node N_i represents a requirement at a given level of detail $level_i$. It can be static (independent of the time) *i.e.* considered verifiable at any time (the set of moment T is empty). It can be dynamic i.e. having to be verified only at some phases of the collaboration life cycle. The $relation_i$ is the refinement relation linking N_i to a set of nodes from $level_{i+1}$ i.e. nodes representing more precise requirements. Last, the $relation_i$ is conditioned by both logical functions θe and θc. θe is the logical function describing the condition in which the requirement is satisfied. θc is the logical function allowing to interpret the influence of the value of sources nodes on the target node N_i.

  - $fact_i$ = {variables, parameters and predicates values extracted from $processModel_i$}
  - $level_i$ ∈ [0 ;+∞[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.
  - $value_i$ ∈ {0, 1} is the result of verification node, *in absentia* 0 (false). With $value_i = \theta e \wedge \theta c$.
- ∃ ! N_0 = ($name_0$, $description_0$, $relation_0$, $fact_0$, $level_0$=0, $value_0$) ∈ N is the root node of the graph G representing the more abstract interoperability requirement.

Where, by convention:

m = number of nodes of the oriented graph G.

n = number of links of the oriented graph G.

$\boldsymbol{T}$ = set of moments.

processModel is the pointed out model of the collaborative process to be analysed.

This model is applied on interoperability requirements reference repository and illustrated Fig. 1. The root node (in gray) is refined into three sub-nodes, each ones represents the three categories of requirements. Each category is then refined with the introduction of new sub-nodes according to the concepts introduced in the enterprise interoperability framework (interoperability concerns and interoperability barriers) (INTEROP, 2007). Each category is refined by

interoperability concerns which are refined themselves by interoperability barriers. The causal tree is then completed with the more precise requirements extracted from several research works about interoperability such as maturity models (Tolk et al., 2003), (C4ISR, 1998), (Clark et al., 1999), (ATHENA, 2005) and an investigation made from enterprises to collect their interoperability requirements. Therefore, possible enrichments with new interoperability requirements are made on the last levels of abstraction in the causal tree.

The choice of logical function (θe) used to link an abstract requirement to more precise requirements is left to user's discretion. Thus, the requirements of each level can be analyzed separately. In this case, some of requirements that are not satisfied, can be considered as negligible or represent an acceptable risk for the user.

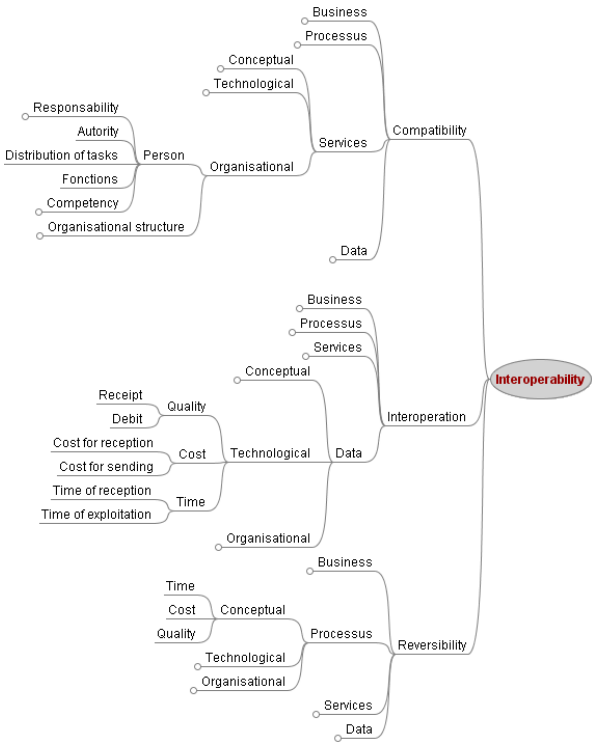

Fig. 1. Interoperability requirements reference repository represented as a causal tree

To analyse the collaborative process, the verification of more precise interoperability requirements by formal verification techniques is then proposed.

3. INTEROPERABILITY REQUIREMENTS ANALYSIS

Different mechanisms are highlighted to analyse interoperability requirements on a model of collaborative process. These mechanisms are based on formal verification techniques (Edmund *et al.*, 1999) (Berard *et al.*, 2001) or on technical expertise.

Indeed, some requirements can be verified on the collaborative process model (*processModel)* - through formal verification techniques - if the modelling language used to build *processModel* allows the description of interoperability requirements. These interoperability requirements can be verified after an adequate translation of the *processModel* into formal models upon which formal verification can be done. In other cases, if interoperability requirements highlight particular points of view of the process and cannot be described by the modelling language, technical expertise is required; this stage of checking is not detailed in this paper.

Interoperability requirements can be verified from formal manner using two verification techniques. The first one is based on conceptual graphs (Chein *et al.* 1992), (Roque *et al.*, 2009) for the verification of static requirements and, the second one is based on model checking for the dynamic requirements (Behrmann et al., 2004).

Formal verification of dynamic requirements (*i.e.* interoperation requirements) is perform with the model checker tool UPPAAL (Behrmann *et al.*, 2004) for different reasons (richness of TCTL temporal logic, open source, user friendly, stand alone tool, ...). The principle of a model checker is to verify properties (that represent requirements formally) exhaustively with temporized and eventually constrained automata that describe the behaviour of the system. A model checker replies with true or false if a property is satisfied or not and, in the second case, gives a counter example.

Verification with model checkers requires two phases. The first phase consists to define an (set of) equivalent behavioural model(s) of the *processModel* and to define the *processModel* transformation rules to be applied. The second phase consists to reformulate the interoperation requirements gathered into the repository under the form of properties and respecting the formal language adopted by the chosen model checker (*e.g.* a temporal logic) (Schnoebelen, 2002). The two phases for the verification of interoperation requirements by the model checker UPPAAL are presented in the next sections.

4. BEHAVIOURAL MODEL GENERATION

The modelling language used to describe *processModel* is BPMN (Business Process Modelling Notation) (BPMN, 2009). It provides a standardised notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible to implement the technology that will perform these processes, and finally, to the business people who will manage and monitor these processes. However, this notation does not consider the description of interoperability such as the nature of the exchanged flow (information, energy, material and person), the availability of resources and their aptitudes. As a consequence, this language has been enriched (1) to become able to embed the interoperability requirements model proposed above and, (2) to make their verification possible. Some of these enrichments are presented in (Roque *et al.*, 2009). Then the *processModel* has to be transformed in behavioural models compatible with the model checker UPPAAL.

In UPPAAL, a model is a set of templates representing Networks of Timed Automata, which communicates with

synchronisation (either on the form Expression! for sending or Expression? for receiving message), using channels and syntax like sent/receive. Each template has locations and transitions to link a location source to a target source (Behrmann *et al.*, 2004).

The proposed transformation is based on (Gruhn *et al.*, 2005) and proposes the transformation of the few BPMN elements: Start and End event, the Gateway (AND and XOR) and the Task (Activity) into templates. For instance, the Start event can be transformed into a simplified template with two locations and a synchronisation as presented Fig. 2. Furthermore, the Task is transformed using four locations and two synchronisations. To consider the message flow between two Activities, another synchronisation between them (message) are added. Moreover, several Start event and End event can be considered using the Declaration System.

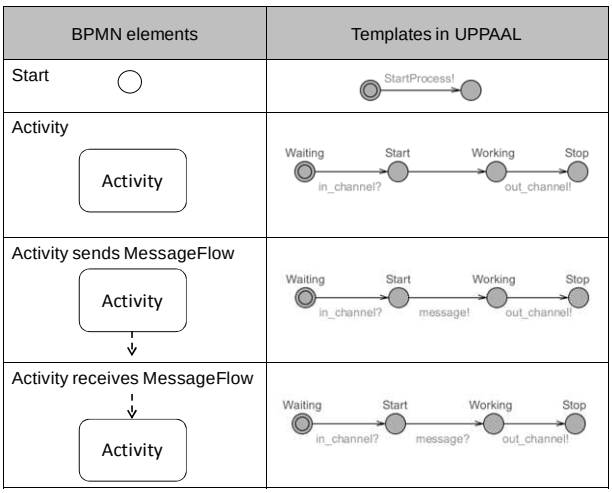

Fig. 2. Examples of transformation rules from enriched BPMN to Networks of Timed Automata

These transformation rules are developed with ATL (Atlas Transformation Language) (ATLAS, 2005) as shown in Fig. 3 in order to re-write the *processModel* into Networks of Timed Automata. The objective is to obtain models without ambiguity in order to check formal properties that describe interoperation requirements.

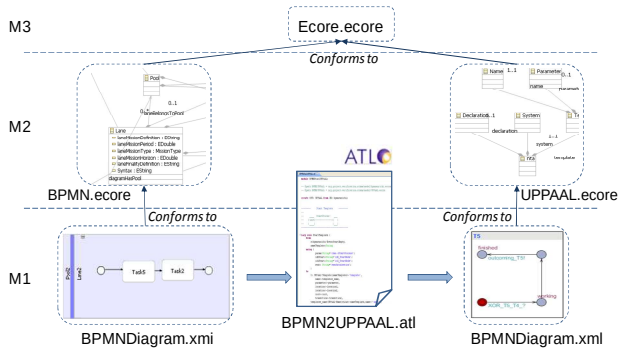

Fig. 3. Transformation from enriched version of BPMN to Networks of Timed Automata

The ATL is a model transformation language specified both as a meta model and as a textual concrete syntax. The main advantage of using ATL is to dispose of two types of model transformation description. The preferred style of transformation writing is declarative, which means simple mappings can be expressed simply. However, imperative constructs are provided so that some mappings too complex to be declaratively handled can still specified. In Fig. 3 the transformation procedure of models (level M1) starts taking into account the meta models (level M2) of the enriched BPMN language and the UPPAAL model checker which are conform to the meta meta model ecore (level M3) of EMF (Eclipse Modelling Framework, available online at: http://www.eclipse.org/modeling/emf/). This transformation is made in order to provide all the needed templates and system declaration of the Networks of Timed Automata. All templates are obtained by considering all the modelling entities which will be used in the checking task. Thus, each class (including its attributes) and each relation of the meta model are translated into templates. Respecting this, each BPMN element can be extracted from the *processModel* in order to produce the corresponding template representing Networks of Timed Automata. Thus, these templates gather all the knowledge described in the model and represents the behavioural model of the collaborative process. At this stage, the ATL transformation remains for the moment unique. The objective is to give the choice of the transformation to the user. For instance, the transformation of task element from the enriched BPMN to a template in Networks Timed Automata can change depending on the wishes of the user. Indeed, a task can be transformed into a simple template which has simple states and two synchronisations using a first ATL transformation and a template which considers resources with another ATL transformation as shown Fig. 4. This choice can influence the requirements checking step.

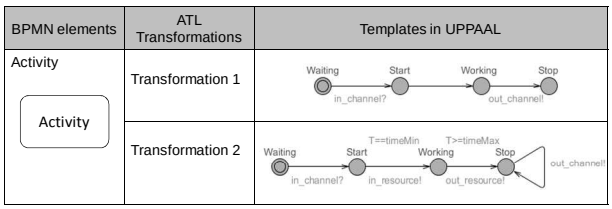

Fig. 4. Exemples of two transformations of an activity from enriched BPMN to Networks of Timed Automata

5. INTEROPERATION REQUIREMENTS FORMALISATION

To enable the implementation of formal verification techniques, the interoperation requirements are formalised into TCTL properties (Timed Computation Tree Logic *i.e.* the UPPAAL property specification language) as presented in next section. TCTL is an extension of CTL (Computational Tree Logic) which allows considering several possible futures from a state of a system. The model checker UPPAAL has four TCTL quantifiers permitting to write the following queries for a property p:

- E<>p: p Reachable *i.e.* it is possible to reach a state in which p is satisfied.
- A<>p: Inevitable p *i.e.* p will inevitable become true.

- E[]p: Potentially Always p *i.e.* p is potentially always true.
- A[]p: Invariantly p *i.e.* p is true in all reachable states.
- p→q: p leads to q *i.e.* if p becomes true, q will inevitably become true.

According to the templates defined above, the interoperation requirements written in natural language are manually re-written into properties using TCTL. Then the model checker UPPAAL verifies exhaustively properties in TCTL through all execution paths of the behavioural models that are reachable. For instance, a requirement described as "*an activity is working between T=5 time units and 10 time units*" can be formalised into a property using TCTL as

E<> Activity.Working and T>5 and T<10

This property indicates that a path can exists where an activity is in the state Working between 5<T<10. This property can be verified on the template representing an Activity shown Fig. 2. To illustrate the proposed approach, examples are given in next section.

6. INTEROPERATION REQUIREMENTS VERIFICATION: APPLICATION CASE

To illustrate the proposed approach, an example showing the progress of four activities in an enterprise is presented. For instance, modelling a collaborative process where several activities are involved can be done in several ways according to several scenarios as shown Fig. 5.

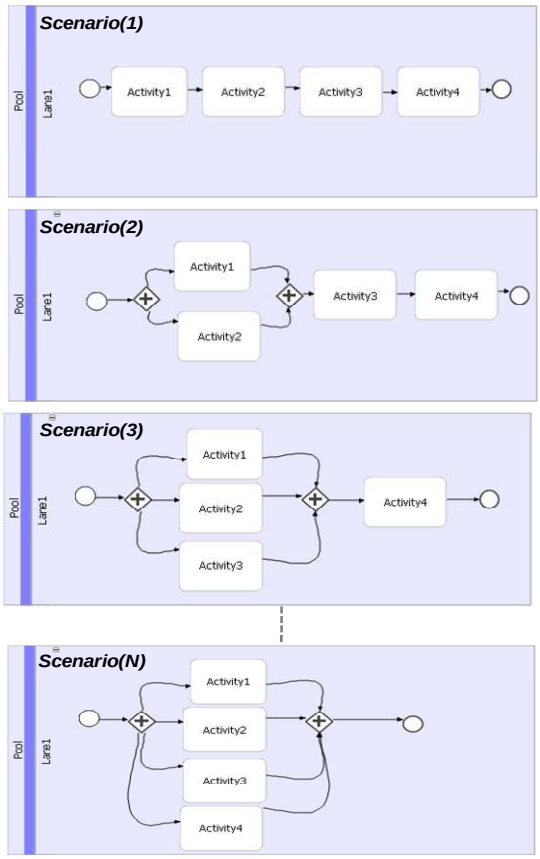

Fig. 5. Several scenarios from enriched BPMN

The first scenario is a sequence of four activities, the second one represents the implementation of the first two activities in parallel and the last one represents the implementation of the four activities in parallel. The objective is to demonstrate that the verification of interoperation requirements can guide the user in selecting the most appropriate scenario according to its needs for the implementation of the collaborative process.

Several interoperation requirements depending on time can be verified on these scenarios.

The interoperation requirement described by "*the resource is available for all activities*" is formalised by the property described as:

E<> Resource.Available and Activity.Start

Where Resource and Activity represent the name of the template and Available and Start represent the name of the state. This property indicates that the resource is in the state Available when the activity starts. To verify this property for the previous scenarios, two templates are needed, a template representing the activity and a template representing the resource as shown Fig 6. The verification of the property will go through all possible paths and answering true or false.

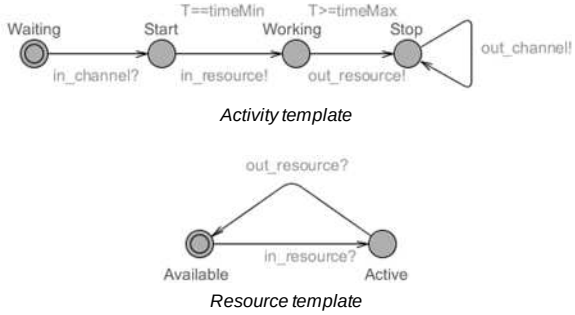

Fig. 6. Activity template and resource template

If two activities (for example Activity1 and Activity4) of the process have to use the same resource, the property is not satisfied if the two activities are in parallel. Indeed, the two activities can use the same resource, only if they are in sequence. The resource can be allocated successively to the two activities. In this case, the Nth scenario do not respect the property (*i.e.* the requirement).

Moreover, this requirement may be verified in a different way. Indeed, if a time condition is added to this requirement, the verification will be different for the scenarios. The interoperation requirement will be described as: "*the resource is available for all activities on time*". For instance, if the first activity uses the resource in the first 10 time units and the second activity needs this resource before the end on the first one at five time units, then the properties to verify on the Networks Timed Automata will be given by:

E<> Resource.Available and T<10 and Activity1.Start"

for the first activity and by:

E<> Resource.Active and T>5 and Activity2.Working

for the second activity where T represents a clock. In this case, the first property is satisfied and the second one is not satisfied. Indeed, if the resource is used by the first activity during 10 time units the second activity cannot access the same resource at 5 time units.

As a consequence, this approach can help the user to detect different problems in the different scenarios. So, the user can choose the scenario which seems the more efficient for the collaboration (to be left to the user's discretion) even if this scenario does not check all the requirements. Indeed, the consideration of the temporal aspect of collaboration changes the result of the verification and can bring more information about the choice of the scenario by users.

7. CONCLUSION

In a collaborative context, interoperability takes a preponderant part. Therefore, enterprises aim to find their interoperability problems and resolve them to have an efficient collaboration. As a consequence, formalisation, and verification of interoperability requirements to help enterprises to find their interoperability problems can be a potential solution to improve this collaboration.

This paper presented the definitions of interoperability requirements and their formulation thanks to a reference repository. This reference repository is used to target interoperability problems in order to solve them easier.

This paper focused on the dynamic aspect of the collaboration, with the verification of interoperation requirements involved during the collaboration. To make the verification of these requirements thanks to a model checker, they must be formalised into properties using a formal language. Future works are related to the verification of interoperability properties with formal verification techniques using multiple kinds of transformations.

REFERENCES

ATHENA. 2005. Framework for the establishment and management methodology, *Integrated Project ATHENA*, deliverable A1.4, 2005.

ATLAS Groupe INA & INRIA Nantes. 2005. ATL Atlas Transformation Language. Specification of the ATL Virtual Machine. Version 0.1. 2005.

Aubert B., Dussart A. 2002. Système d'Information Inter-Organisationnel, *Report Bourgogne, CIRANO*, March 2002. (in french)

Behrmann G., David A., Larsen K. G., 2004. A tutorial on Uppaal Department of Computer Science, Aalborg University, Denmark, 2004. Available online at : http://www.uppaal.com/ (last visited: 04-04-2011)

Bérard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen Ph., McKenzie P. 2001. Systems and Software verification: model checking techniques and tools, Springer, 2001.

BPMN. 2009. Business Process Modeling Notation, V1.2. http://www.bpmn.org/, 2009 (last visited: 04-04-2011)

Camarinha-Matos, L. M., Afsarmanesh, H., Rabelo, R. J. 2003. Infrastructure developments for agile virtual enterprises. *Journal of Computer Integrated Manufacturing*, ISSN 0951-192X, Vol. 16, N. 4-5, Jun-August 2003

Chapurlat, V., Roque, M. 2009. Interoperability constraints and requirements formal modelling and checking framework. *Advances in Production Management Systems*, APMS, Bordeaux, France, 2009

Chein, M,, Mugnier, M-L, 1992. Conceptual graphs: fundamental notions, *Revue d'intelligence artificielle*, vol.6, n°4, pp. 365-406, 1992

Clark, T., Jones, R. 1999. Organisational Interoperability Maturity Model for C2. *Proc of Command and Control Research & Techn. Symposium*, Newport, USA 1999

C4ISR Architecture Working Group. 1998. Levels of Information Systems Interoperability (LISI). *United States of America Department of Defence*, Washington DC, USA, 30 March 1998

Daclin N., Chen D., Vallespir B. 2008. Methodology for enterprise interoperability. *17th IFAC World Congress* (IFAC'08), Seoul, Korea, 2008.

Edmund M. Clarke Jr. 1999. Orna Grumbereg, Doron A. Peled, Model checking, *The MIT Press*, 1999.

Gruhn V., Laue R., 2005. Using Timed Model Checking for Verifying Workflows. *Computer Supported Activity Coordination 2005*: 75-88.

INTEROP. 2007. Enterprise Interoperability-Framework and knowledge corpus - Final report, *INTEROP NoE, FP6 – Contract n° 508011, Deliverable DI.3*, May 21st

ISO/DIS 11345-1, 2009. Advanced automation technologies and their applications. Part 1: Framework for enterprise interoperability. 2009

Mallek S., Daclin N., Chapurlat V. 2010. Toward a conceptualisation of interoperability requirepments. *IESA 2010, Interoperability for Enterprise Software & Applications*, Coventry, 14-15 April 2010

Roque M., Chapurlat V. 2009. Interoperability in collaborative processes: requirements characterisation and proof approach, *PRO-VE'09, 10th IFIP Working Conference on VIRTUAL ENTERPRISES*, Thessaloniki, Greece, 7-9 October 2009.

Schnoebelen Ph. 2002. The Complexity of Temporal Logic Model Checking, *Advances in Modal Logic*, Volume 4, pp1-44, 2002.

Scucanec, S. J., Van Gaasbeek, J. R. 2008. A day in the life of a verification requirement. *U.S Air Force T&E Days*, Los Angeles, California, February 2008

Tolk, A., Muguira, J.A. 2003. The Levels of Conceptual Interoperability Model. *Proceedings of Fall Simulation Interoperability Workshop (SIW)*, Orlando, USA, (2003)