Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

*

A Systemic approach to Interoperability formalization

Yannick Naudet * **Thibaud Latour** * **David Chen** **

* *CITI, Henri Tudor Public Research Center, 29, Avenue J.F. Kennedy, L-1855 Luxembourg - Kirchberg, Luxembourg*
** *IMS/LAPS, University Bordeaux 1, 351, Cours de la liberation, 33405 Talence cedex, France*

Abstract: With a first version developed last year, the Ontology of Interoperability (OoI) aims at formally describing concepts relating to problems and solutions in the domain of interoperability. From the beginning, the OoI has its foundations in the systemic theory and addresses interoperability from the general point of view of a system, whether it is composed by other systems (systems-of-systems) or not. In this paper, we present the last OoI focusing on the systemic approach. We then integrate a classification of interoperability knowledge provided by the Framework for Enterprise Interoperability. This way, we contextualize the OoI with a specific vocabulary to the enterprise domain, where solutions to interoperability problems are characterized according to interoperability approaches defined in the ISO 14258 and both solutions and problems can be localized into enterprises levels and characterized by interoperability levels, as defined in the European Interoperability Framework.

1. INTRODUCTION

Interoperability is a requirement inside a system, which maturity depends on the kind of existing interactions or composition among its components. It is the same for the system itself, when it needs to be sufficiently flexible to interact with another system, or if it needs to be open to new components. As soon as this ability is not achieved when systems or system's elements need to operate together, interoperability becomes a problem that must be solved. These assertions stay valid for whatever kind of system. The general perspective provided by such a "system" approach is especially important when considering the different facets of interoperability, like defined *e.g.* in the European Interoperability Framework (see CompTIA [2004]): technical, organizational, and conceptual (semantic). Indeed, we can take the same general approach and reasoning whatever the considered system, be it technical (*e.g.* IT systems), organizational or conceptual (e.g. models), or a composition of these. More generally, a system can be real or abstract and can belong to any part of another system: a software component, as well as a hardware component, a person, a business rule, etc., can all be elements of a system.

The Ontology of Interoperability (OoI) aims at formally defining interoperability and providing a framework to describe problems and related solutions pertaining to the interoperability domain. Associated with a knowledge base of problem and solution descriptions, it will constitute the basis for a decision-aid system providing interoperability problem detection and identification, and solution proposals. We have first formalized and presented the OoI in Naudet et al. [2006], based on the preliminary work of Rosener et al. [2004] and Rosener et al. [2005]. At this time, it was a conjunction of three models: a *systemic model*, a *resource composition model*, and *a decisional model*. We based it on a definition considering interoperability as a problem caused by a contact between heterogeneous resources in a system, thus taking into account not only behavioural (*e.g.* communicational) interoperability, but also structural (*e.g.* physical). We further refined this view in Ruokolainen et al. [2007], enhancing the ontology with concepts related to the dynamic nature of interoperability: *indicators* for problem detection, *negotiation* and *compensation* solutions, and the notion of *environment*. The latter refined view also introduces the important concept of *incompatibility*, which is broader than *heterogeneity* and slightly changes the ontology.

In parallel to the work on the OoI, D. Chen proposed the Framework for Enterprise Interoperability (FEI) in Chen and Daclin [2005] and Chen [2006], which aims at capturing and structuring interoperability knowledge for enterprises. As for the OoI, this framework starts from the hypothesis that interoperability is a problem of incompatibility, which needs to be solved. The FEI defines interoperability barriers categories and introduces approaches to solve interoperability that are general enough to be considered in the OoI.

This paper first summarizes the work done so far on OoI. It then proposes a new version with foundations strengthened in the systemic theory, and finally a possible integration of the FEI. The reminder of the paper describes the new ontology, highlighting restructuration of the model and the changed concepts. The OoI is now based on two models, namely the *systemic model* that includes the old resource composition model introduced in Rosener et al. [2005], and the *decisional model*. We then discuss the integration of the FEI concepts and finally illustrate with a concrete use case the formalization of an interoperability problem and a possible solution. Using the FEI concepts, the problem is precisely characterized

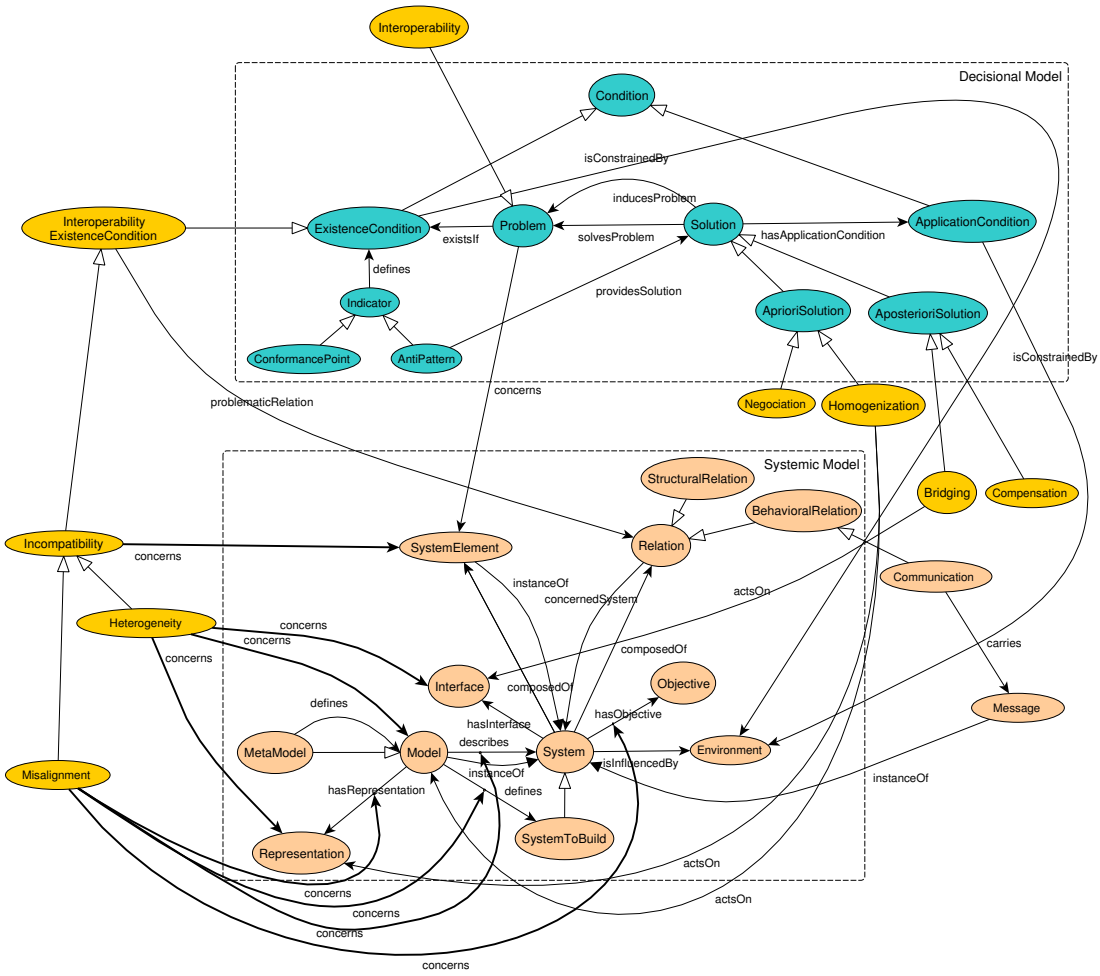

Fig. 1. The Ontology of Interoperability (OoI), enhanced to follow a full systemic approach. The two composing models, namely the *systemic model* and *decisional model*, are highlighted.

by a barrier category, and solution can be related to the interoperability approach it uses.

2. THE ENHANCED SYSTEMIC MODEL

According to the General Systems Theory (Von Bertalanffy [1968]), a system is defined as a set of interconnected parts, having properties that are richer than the sum of the parts' properties. A system can be characterized by the elements composing it, which can themselves be organized in sub-systems potentially interconnected. But the most important of its components are the relationships between elements: because they exist, the system possesses its own (emerging) properties. A system has boundaries, locating it in the super-system it belongs to, and a set of specific attributes characterizing what it is and differentiating it from another system.

Starting from the general point of view of a system and a pragmatic problem-solving approach, we consider interoperability as being a problem, which can arise only when some resources are put together in one system and need to inter-operate. As such resources of a system are themselves systems, interoperability simply concerns relations between systems. In this paper, we adopt a system-oriented definition of interoperability, which is a generalization of the one we adopted before in Rosener et al. [2005]:
An interoperability *problem* appears when two or more *incompatible* systems are *put in relation*. Interoperability *per se* is the paradigm where an interoperability problem occurs.

The current version of the OoI is presented in figure 1. In the bottom part, the enhanced systemic model describes systems, no more as a composition of inter-related resources like in the former OoI version, but of inter-related systems. The reason for having done this is simple: resources are elements of a system and systems themselves. We now use the term *SystemElement* instead of *Resource*, which is also more precise as its implicit semantics highlights the part-of relation existing with the super-system. A *System* instance is composed by *SystemElement* instances, which are systems themselves, and *Relation* instances. This simple view of a system allows inherently to cope with systems of systems, whatever the kind of systems in relation. The *Relation* class formalizes the existing relationships inside a system, which is of primary importance in the systemic view and is at the basis of the occurrence of interoperability problems. As stated in Walliser [1977], relations can be local or global, *i.e.* between elements of a system or between the system and

its sub-elements. As a consequence, *Relation* is directly linked to *System*, by the *concernedSystem* property.

The systemic model defines the central concepts of *System* and *Model*. *System* is further specialized in *SystemToBuild* in order to make the difference between an existing system and a non-existing one that will have to be built. In engineering, a system is the result of a building process, resulting itself in a system, built from an a priori model. Such model is usually established from a given perception of the reality, which is also at the basis for building models for existing systems. From this observation, it appears that system and model relations are dual: on the one hand, some models describe existing systems with the objective of understanding the reality; while on the other hand, some models define a new system to build. In the OoI, the *Model* and *System* classes can be related by either the *describes*, or the *defines* property. The former is not linked to *SystemToBuild* because even if it is primarily used at system building time, the relation persists after the system has been build and thus concerns systems in general. A model can also be a *MetaModel*, used to define a model. We also define the *Representation* of a model, which is a system constituting a perceivable symbolic description of this model. Models' representations will be of particular importance when discussing syntax issues in interoperability.

According to the systemic approach, Le Moigne defines a canonical form of the general system as being: a set of *actions* in a specific active *environment*, in which it *functions* and *transforms*, for a specific *objective* (see Le Moigne [1990]). In the OoI, the *System* class is linked to an *Objective* and an *Environment*. The objective represents what the system is meant for, what are its goals, how it can be used, what is its behavior, etc. An inter-operation conflicting with an objective is a case of interoperability problem. The environment represents all that is external to a system but can have an influence on it, as it contains the system. It imposes constraints on a system, but also on the existence of a potential problem and application conditions of a solution. The three other characteristics of a system in Le Moigne's definition, can also be assimilated to a system's structure and its behavior. In the OoI, the model and the composing sub-systems and relations of a system represent these characteristics. From an interoperability point of view, the hypothesis of quasi-isolated systems (black-boxes) is often taken. When two or more systems need to inter-operate, the *Interface* they provide to communicate with the external world is often the only thing considered. This is especially true in Software Engineering. With this approach, the system behavior (its running actions and transformations) is mostly reflected through the interface, which we assimilate to the set of inputs/outputs characterizing quasi-isolated systems. We will see in section 4 that this is not always sufficient and that going deeper by considering the system's model or internal structure can be necessary.

3. THE DECISIONAL MODEL

The decisional model presented in the top part of figure 1 is designed within a problem-solving perspective. This model aims at being the main support for a decision-aid system suggesting solutions to identified problems. It is designed with the objectives of 1) Separating the problem from the solution, enforcing prior analysis of the problem before suggesting any solution; 2) modeling problems independently in a reductionist view to reduce the decision complexity; and 3) Acknowledging that a solution of any kind will be the source of new problems, which recursively have to be solved. The third point contributes to the realization of the second one by providing a way not to consider composite problems: relations between problems are only seen indirectly through problems induced by solutions applied to them, allowing a recursive approach for resolution.

The main concepts of the decisional model are those of *Problem* and *Solution*. Problems concern the elements of a system (*SystemElement* in the systemic model). Solutions *solve* problems and can in turn *induce* new problems. Any problem or solution being valid in a particular context or environment, conditions of respectively existence and application are of primary importance. While the applicability of solutions might be dependent from *e.g.* technical infrastructure, cost considerations or organizational factors, problems arising in a given situation might not appear in a different one. In the model, the *Condition* class is specialized into *ExistenceCondition* and *ApplicationCondition*, respectively linked to *Problem* and *Solution*. Both are related to the *Environment* class in the systemic model, with a *isConstrainedBy* property.

Problems and solutions are often related by a temporal entailment. From the systemic model, a system is either to understand or to build, and the model used to describe a system always comes before the one defining the new system. If the point of reference is the building process of a new system, according to circumstances, one may either solve problems before building the system, or after the system has been built. Solutions solving problems by anticipation are instances of *AprioriSolution*, and *AposterioriSolution* can be used for solutions correcting problems after they occurred. Both classes are specializations of *Solution*.

While the described concepts and relations constitutes its core and are sufficient for decision making, the decisional model has been enhanced, in particular to help dynamic detection, at the level of existence conditions of problems. These ones can be formally defined as *Indicator*s, further declined into *anti-patterns* and *conformance points*, which both aims at being checked at runtime to detect the occurrence of potential conflicts. Anti-patterns and conformance points are dual indicators: the verification of the first one indicates the existence of a problem, while the last one indicates a problem if it is verified as being false. Conformance points provide a description of checking points (*e.g.* in the form of a set of rule) that must be verified to test the correspondence between the expected behavior and actual operation of the system; while anti-patterns describe specific knowledge or bad solutions or habits leading to problems. The interesting aspect of anti-patterns is also to provide a solution to avoid the problem they relate.

Interoperability can be modeled based on systemic and decisional models, on which we link classes and properties

specific to the interoperability domain. The next section presents these classes and properties in the OoI.

4. THE ONTOLOGY OF INTEROPERABILITY

The Ontology of Interoperability, presented in in figure 1, is formalized on the basis of the definition provided in introduction and the two models presented in the preceding sections. Interoperability is then logically implemented as a subclass of the *Problem* concept. Problems of interoperability exists when there is a relation, of any kind, between incompatible systems in a super-system they belong to or they will form (*i.e.* a system to build). Since the latest revision of our ontology, presented in Ruokolainen et al. [2007], the considered scope of interoperability has been extended to that of incompatibilities between systems. Previously defined as the condition of existence for interoperability problems, the *Incompatibility* concept is now a subclass of a more generic *InteroperabilityExistenceCondition* class aiming at explicitly formalizing the fact that a relation is an existence condition for interoperability problems. The *Incompatibility* class is further partitioned into *Heterogeneity* and *Misalignment*. Incompatibility is the source of interoperability problems for systems of any nature, as soon as they belong to the same super-system and there exist a relation of any kind between those systems: a simple communication or contact link, a constraint imposed on a system by another, an influence or perturbation, etc. *Incompatibility* is thus related to *SystemElement*. Linking to *System* would have been too general and would have not represented the fact that interoperability is concerned by inter-related systems, in the context of a super-system: incompatibilities of systems that do not have to interoperate is out of the scope.

Then, heterogeneity relates to systems of the same nature and concerns either their *interfaces*, their *models*, or the *representations* of their models. Detecting heterogeneity problems implies considering these three points. A first 'surface' check can be performed on the interfaces to detect apparent problems. When interfaces are homogeneous, there is no interoperability problem whatever the internal structure of the systems intended to inter-operate. Heterogeneous interfaces constitute the most commonly considered interoperability problems. Common cases are *e.g.* method signature of a library not compatible with the calling program, diameter of the screw different from the diameter of the nut, etc. A second check concerns systems' internal structure. Heterogeneous models lead to behavior incompatibilities. When building a system, having elements with different models is also problematic (for instance when nut and screw are made of plastic and steel). Heterogeneity of meta-models if often related to semantics, while syntactic heterogeneities concern models' representations.

Misalignment can be observed when a system constraints the way another one will be build, structured or will behave. The two systems are in this case of different nature. Misalignment does not occur between the systems themselves, but at the level of the relations: between a system and its model (*e.g.* constraint on the structure); between a system and its representation (*e.g.* constraint on the syntax of the model); between a system and its objective (*e.g.* constraint on the behavior).

We have modeled two types of solutions to interoperability problems that can be applied respectively at the technical or organizational level. From a pure technical point of view, *Bridging* and *Homogenization* seems to be the only alternatives. Models for bridging and homogenization have been proposed in Rosener et al. [2004] and have not evolve since then. We however explain them here. At the organizational level, *Compensation* or *Negotiation* can be used.

Homogenization is an a priori solution to an heterogeneity problem, acting on models or their representations. It requires two basic applicability conditions to be effective: first, the modification of the concerned systems must be possible, and second, one must have a sufficient knowledge about the system's components to homogenize. Such knowledge is contained in the models that have been used to build the systems. Those models will have to be modified so that a new homogeneous system can be rebuild from the original inter-related systems. Homogenization requires *transformations*, which can be either *syntactic* or *semantic*. Such transformations are performed using a *unified model*, which can be of several kinds: a *unified language* to achieve homogenization ; a *unified meta-model* (or ontology) to reduce semantic heterogeneity; or a *unified interface*.

If homogenization is generally the preferred solution to ensure a good validation of the resulting system, this solution is often hardly feasible due to lack of control on the legacy system preventing one from modifying it and/or lack of models. Moreover, this solution induces a series of problems that needs to be considered: the *validation of the unified model* and *verification of transformations*, the errors generated by the *modification of systems*, and *changes in performance*.

When homogenization is not possible or when it creates more problems than it can solve, bridging is the alternative. Bridging consists in inserting a new system that will serve as an intermediate between inter-related systems causing an interoperability problem. It is an a posteriori solution to an heterogeneity problem, acting at different levels in the systemic model. We call the bridge system an *adapter*, and it often relies on a *translation protocol* to actually transfer information from and to interacting systems. Typical examples of bridging solutions in IT are Enterprise Application Integration, or a plug adapter between *e.g.* DIN and US standards (structural relation).

Bridging solutions induce a series of problems related to *performance* and *integrity*. Performance problems are mainly due to the translation process that take place in the bridging solution. This translation is resource consuming and requires a careful verification of requirements in terms of performance and response time for the whole system. Moreover, the new added component may introduce alterations to *robustness*, *security* or *safety* of the system.

At the organizational level, *Negotiation* mechanisms are used when different actors influencing different systems have different views, objectives or business processes that must be taken into account when trying to build an interoperable system. Negotiation allows finding mutual agreements between actors. When the negotiation process fails at system building time, an alternative is to use some

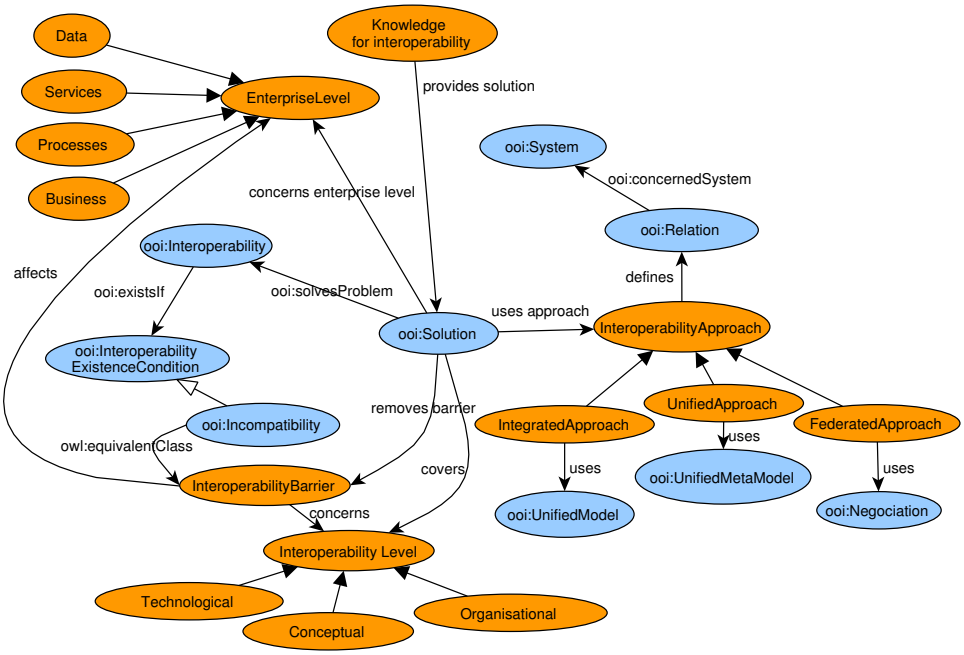

Fig. 2. Representation of the FEI, integrated into the OoI. The concepts of the OoI are identified by the *ooi:* prefix.

kind of *Compensation* a posteriori: *i.e.* an actor accepts something in exchange of a disagreement, for the fulfilment of the super-system objective.

5. INTEGRATING THE FEI

The Framework for Enterprise Interoperability (FEI), which has been proposed in Chen and Daclin [2005], defines a multidimensional classification scheme for categorizing knowledge for interoperability. In this section, we present its integration in the OoI. It adds a specific vocabulary specializing the OoI to the enterprise domain, allowing knowledge for interoperability to be described as providing one or more solutions to solve interoperability problems. These solutions and problems can be localized by referring to enterprise levels and are directly linked to interoperability levels as defined in the EIF (CompTIA [2004]). Moreover, the inclusion of interoperability approaches defined by the ISO 1458 (integrated, federated and unified) allows to define the way the relations between systems that caused the problems should be.

The FEI is represented by three main dimensions that we represent by the three concepts: *InteroperabilityBarrier*, *InteroperabilityApproach*, and *EnterpriseLevel*. They are all modeled with their different constituents represented here as instances. As these main concepts provide a way to classify or characterize solutions for interoperability we can link them to the *ooi:Solution* class. The three dimensions are then respectively related to solutions by *removes barrier*, *uses approach*, and *concerns enterprise level*. According to Chen [2006], interoperability approaches are ways to relate entities (systems) together to establish inter-operation. This is modeled with a link between *ooi:Relation* and *InteroperabilityApproach*. Basically, a relation concerns at least two systems, and an *InteroperabilityApproach* instance will define a relation, as the choice of one or another such approach will actually influence or constrain the way the relation between the systems will be build so as to make them inter-operate. In Chen and Daclin [2005], the term barrier is defined as an incompatibility, obstructing the sharing of information and preventing exchanging services. By assimilating it (with the owl:equivalentClass in figure 2) to the *ooi:Incompatibility* concept, we give *InteroperabilityBarrier* a broader sense, extending it to any kind of incompatibility, obstructing a correct communication, connection or behavior of the concerned systems. Last, the full integration of the FEI into the OoI implies considering the *Knowledge for Interoperability* concept, representing pieces of knowledge like *e.g.* PSL relevant to interoperability (see Chen [2006]). Knowledge for interoperability instances provide solutions to remove barriers (*removes barrier* link to *InteroperabilityBarrier*) at a particular enterprise level (*concerns enterprise level* link to *EnterpriseLevel*), through a specific interoperability approach (*uses approach* link to *InteroperabilityApproach*).

The previous paragraph explain the inclusion of the FEI as it has been originally defined. However, it seems to us that it could be slightly changed to better fit the goals of the OoI that are to serve as a meta-model to describe problems and solutions pertaining to interoperability. Indeed, if solutions for interoperability can be classified according to the three cited dimensions, barriers also can be characterized according to two of them. They obviously affect enterprises levels, and concern specific interoperability levels. If we decouple these levels from the barriers, we obtain two dimensions along which both problems (through barriers) and solutions can be localized: the interoperability level and the enterprise level. The interoperability approach remains linked to solutions only. The resulting integration, with the inclusion of the *Interoperability level* concept and associated links is presented in figure 2. An *InteroperabilityBarrier concerns* an *Interoperability level*, and Solutions can be now described according to the *Interoperability-*

Barrier they remove (*removes barrier* property) or the *Interoperability level* they cover (*covers* link).

6. ILLUSTRATIVE USE CASE

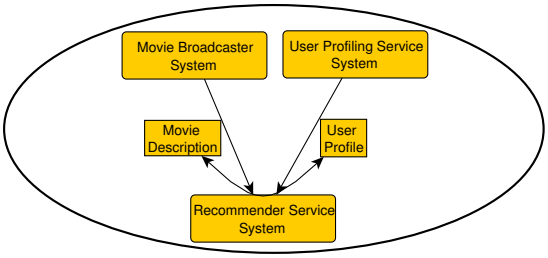

Fig. 3. The super-system formed by a Broadcaster system and a User Profiling system both in relation with a Recommender system.

This section describes a concrete use case for which we show the reasoning process leading to the detection and localization of an interoperability problem, using the OoI. Assume a mobile service company C_A, which acquires a small company C_B offering an Internet forum service where users share their preferences and feelings concerning movies. C_B possesses also a user profiling service which C_A aims at reusing for building its own recommendation service for broadcasted movies. Recommendation is to be made on movies coming from a broadcaster C_C. Technically, C_C provides movies described in MPEG-7 (XML), while C_B's service constructs user profiles in OWL/XML. When implementing its recommender, C_A will face an interoperability problem as it needs to perform matchmaking on data encoded using different languages.

The use case can be represented in terms of systems as in figure 3. The recommender service of C_A is a *system to build*, in which the user profile UP and a movie description MD will be compared and thus put in relation. C_A's recommender (C_AR), C_B's user profiling service (C_BUPS) and C_C movies broadcasting service (C_CMBS) form together a super system in which the interoperability problem occurs. The *relations* $R(C_BUPS, C_AR)$ and $R(C_CMBS, C_AR)$ are not problematic: both relations are simply data transmission to C_AR and there is no direct relation between C_BUPS and C_CMBS. Both relations are of type *communication*, which carries on *messages* containing repectively UP and MD, put in relation by C_AR. This relation $R(UP, MD)$ is where an *heterogeneity* can be detected, not at the level of *interfaces* because there is no direct "contact", but at the level of *models*. UP and MD are indeed *models*, which *describe* respectively a *user*, and a *movie*. The representation of UP and MD are both XML and thus are not the cause of any problem. But UP is described by a model, which is a *user ontology* $UOnto$ expressed in OWL (being itself the descriptive model of $UOnto$). MD being described by a model, which is MPEG7, there is an incompatibility here.

Assuming there exists a knowledge base of known interoperability problems and solutions containing in particular the fact that MPEG-7 and OWL are incompatible, the aforementioned reasoning leads to the detection of the interoperability problem, caused by an *heterogeneity* (or *barrier* in EIF terms) between two systems at the levels of their *meta-models*. This barrier concerns *conceptual interoperability*, and it can be located at the *services* and *data enterprise levels*. Concerning solutions, an *homogenization* can be applied by e.g. using an MPEG-7 ontology $MPEG7Onto$ expressed in OWL. In this *unified approach*, the homogenization at the model level would be finalized by mapping related concepts of $UOnto$ and $MPEG7Onto$. Then, a new MD instantiated from $MPEG7Onto$ would be used for matchmaking with UP in the recommender of C_A.

7. CONCLUSION AND PERSPECTIVES

In this paper, we have presented a refined version of the Ontology of Interoperability, which is more founded in the systemic theory. We have presented the current OoI and discussed an enhanced version of our systemic model. Accounting for the fact that a system is composed by other systems and relationships between them, this model deals inherently with systems of systems. We have moreover included the concepts used in the Framework for Enterprise Interoperability and shown with a concrete use case how they complete the formalization already provided by the OoI by allowing to locate problems and solutions at a specific enterprise level, specifying which interoperability level is concerned and which approach is taken to solve the problem. The example we have given illustrates a typical, fully system-oriented, reasoning process that can be followed using the OoI. The research summarized here will constitute a basis for an upcoming PhD study that will deepen the systemic foundations, investigate the use of metrics and interoperability maturity levels, and build a decision support system based on an enhanced OoI.

REFERENCES

David Chen. Enterprise Interoperability Framework. In *Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI-INTEROP)*, January 2006.

David Chen and Nicolas Daclin. Framework for enterprise interoperability. In *Interoperability for Enterprise Software and Applications, I-ESA05*, pages 77–88, 2005.

CompTIA. European interoperability framework - ict industry recommendations. White paper, CompTIA, Brussels, February, 18 2004.

Jean-Louis Le Moigne. *La modelisation des systemes complexes*. Dunod, 1990.

Yannick Naudet, Thibaud Latour, Kevin Haussmann, Sven Abels, Axel Hahn, and Paul Johannesonn. Describing Interoperability: the OoI Ontology. In *Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI-INTEROP)*, January 2006.

Vincent Rosener, Thibaud Latour, and Eric Dubois. A model-based ontology of the software interoperability problems: preliminary results. In *CAISE04 workshops, EMOI 04*, volume 3, pages 241–252, 2004.

Vincent Rosener, Yannick Naudet, and Thibaud Latour. A model proposal of the interoperability problem. In *CAISE05 workshops, EMOI 05*, volume 2, pages 395–400, 2005.

Toni Ruokolainen, Yannick Naudet, and Thibaud Latour. An ontology of interoperability in inter-enterprise communities. In *Interoperability for Enterprise Software and Applications, I-ESA07*, 2007.

Ludwig Von Bertalanffy. *General System Theory: Foundations, Development, Applications*. Georges Braziller, Inc., New York, USA, 1968.

Bernard Walliser. *Systemes et modeles, introduction critique a l'analyse des systemes*. Editions du Seuil, 1977.