Describing Interoperability: the OoI Ontology

Yannick Naudet[1], Thibaud Latour[1], Kevin Hausmann[2], Sven Abels[2], Axel Hahn[2], and Paul Johannesonn[3]

[1] Henri Tudor Public Research Center, Innovation Center by Information Technologies (CITI), Luxembourg
[2] Business Information Systems, University of Oldenburg, Germany
[3] Royal Institute of Technology (KTH), Stockholm, Sweden

Abstract. Though ontologies are widely used to solve some specific interoperability problems, there is no specific ontology defining what interoperability actually is, independently from any domain. In this paper, we propose and discuss a first version of such an ontology, namely the OoI (Ontology of Interoperability), which we formalized using the Ontology Web Language (OWL). On the basis of previous research efforts having lead to UML formalization of our model of Interoperability, we use this paper for presenting the OWL version and for linking and comparing it with other models dealing with Interoperability: maturity models for interoperability like e.g. the Levels of Information System Interoperability (LISI) model, and the Model Morphisms ontology (MoMo), which deals with interoperability of models. Finally, we illustrate in a brief use case how the OoI could be used with MoMo to provide solutions to interoperability problems between two models.

1 Introduction

Interoperability is commonly implicitly seen as an objective to reach when designing complex systems. Regarding to the IEEE association interoperability is *the ability of two or more systems or components to exchange information and to use the information that has been exchanged* [5]. Considering not only information transmission aspects, interoperability is a requirement inside a system for allowing interaction or composition of its components, but also for the system itself, when it needs to be sufficiently flexible to exchange information with another system, or if it needs to be open to new components. This is the basis of systems interoperability maturity models like the Level of Information Systems Interoperability (LISI) model [4] that we discuss further in section 3.2. As soon as this ability is not achieved when systems or system's components need to operate together, interoperability becomes a problem that must be solved.

A system wide interoperability is only possible if all subsystems are able to work together. This means that all underlying interoperability problems have to be solved. According to [9] an interoperability problem is *the problem of bridging together heterogeneous and distributed information resources and services.* As for the IEEE definition, and other ones that we already discussed in [10], this

definition however covers only one aspect of interoperability. We try to provide a larger vision with our own definition, which we recall in the following section.

Because a simple definition is not sufficient, we try to develop a formal model of the interoperability problem. The so-called Ontology of Interoperability (OoI) aims at providing a formal (and thus processable) definition of interoperability, but also to provide a way to describe interoperability problems as well as possible solutions. This may on the one hand help to get a homogeneous description of various interoperability issues. On the other hand, it may be used to find similar problems and hence to detect solutions that might help to solve a concrete problem. The work presented here has been conducted in the context of the FP6-IST INTEROP Network of Excellence (NoE) Joint Research Activities [6]. In addition to presenting the first version of the OoI, we provide here the first investigations toward a validation of our model. The points addressed here are the following: the connections with other related models or standards, and the illustration of the use of the model on concrete interoperability problems.

The rest of this paper is structured as follows. First, we present the current state of the Ontology of Interoperability (OoI). We then explore links with other models or efforts concerning Interoperability modelisation. We briefly explain existing links with the interoperability Glossary [9], an effort from another task group of the INTEROP NoE. Then, some interesting Interoperability models are presented and compared to the OoI, as well as derived facts and conclusions. We present then the Model Morphisms (MoMo) ontology [3], which can be used in association with the OoI to solve interoperability problems between models. Section 4 presents a brief use case illustrating how the OoI and MoMo provide interoperability solutions between two standard models. Finally we conclude by discussing the current state of the ontology as well as future work.

2 OoI as a model for Interoperability

It is our objective to propose a formal and general model for the definition of Interoperability in order to serve as a representation for common understanding. Using a clear problem-solving pragmatic view, the ultimate goal is to define a framework for the creation of Problem-Solution-Induced Problem knowledge base supporting decision in the domain of interoperability.

Interoperability is a problem pertaining to systems of any sorts. As this concept is often associated to information systems, it is also very often considered as communication issues between or inside systems. However, beyond software engineering and computer science in general, interoperability problems can also occur between 'hardware' components. Starting from a systemic point of view, we do not restrict the scope to information system, but rather consider a system as a *group of independent but interrelated elements comprising a unified whole*[4]. Generally speaking, the components of a system do not necessary have to communicate, but might simply have to be composed together for a specific purpose.

[4] WordNet 2.1 Copyright 2005 by Princeton University

From a pure compositional point of view, this can be viewed as a *structural interoperability* need. When communication or other kinds of action define the relation between the system's components, this concerns the *behavioral aspect of interoperability*. In our view, interoperability is not only a matter of communication as it does not necessarily imply a behavioral aspect. According to this, we propose the following definition:

Definition 1. *An interoperability* problem *appears when two or more* heterogeneous *resources are* put together*. Interoperability* per se *is the paradigm where an interoperability problem occur.*

According to this definition, we created the OoI, which is based on research we have published in [10] and [11]. The center of our ontology is a *Problem*, which might be solved by one or more *Solutions*. We distinguished between an *AprioriSolution* and a *AposterioriSolution*. Another central element is the *System*, which is described by a specific *Model*. An extract of the proposed ontology and its most important elements is displayed in figure 1.

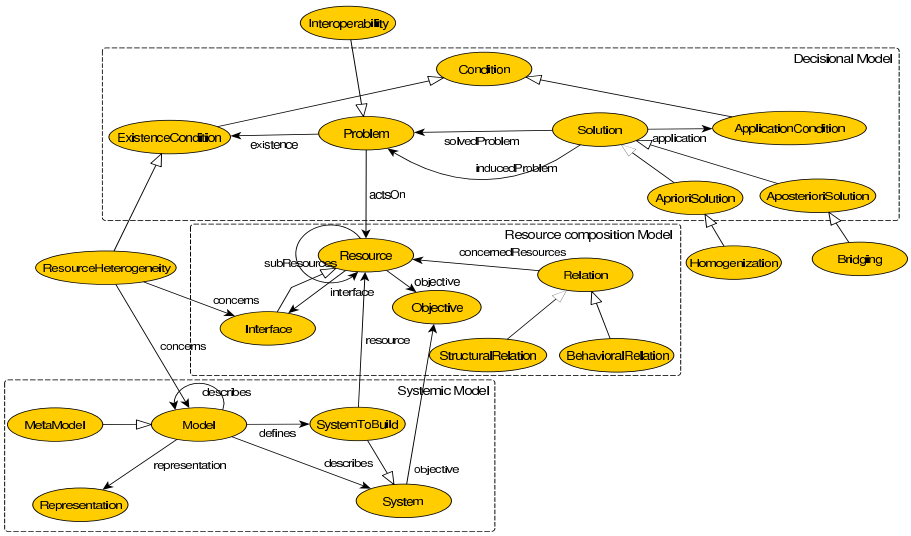

Fig. 1. Extract of the Ontology of Interoperability.

The OoI is structured according to a series of more general models describing the important concepts that appear in the definition: problem and solution, resource composition, and system. Our ontology establishes relations between these three fundamental models. At a conceptual level, these relations are sufficient to provide a definition of what interoperability is. However, at a more operational level, *i.e.* when the ontology is used to structure or describe real cases or if one wishes to set up an interoperability problem-solution knowledge

base, this is not enough. For this reason, we detailed further some of the concepts arising in the definitional part of the model by providing other specific models pertaining to the two classes of solutions: homogenization and bridging, as well as to communication.

In the next part of this section, we shall briefly present each fundamental models followed by the interoperability and more specific models. The systematic definition of the most important concepts and relations of the OoI is out of the scope of this paper. However, the complete OWL file can be obtained from the authors.

2.1 The decisional model

Because interoperability has been defined as a problem, and because we adopted a clear problem-solving position regarding our definition of ontology, the decisional model is at the heart of OoI. This model clearly separates the problem from the solution, enforcing prior analysis of the problem before suggesting any solution. As modelling problems independently in a reductionist view reduces drastically the complexity of the decision, we did not represent problem composition directly. Rather, relations between problems are only seen indirectly through the introduction of new problems induced by the solutions selected to solve other problems. These new induced problems should then be solved recursively.

Both problems and solutions are related to *existence* and *application conditions*, respectively. These are essential, to take into account the context of the decision, *i.e.*, among all possible problems and solutions, the subset of those that are pertinent with respect to the particular situation.

Problems and solutions are often related by a temporal entailment. Indeed, the solution used to solve a particular problem may be applied before the problem effectively occurs, or after it occurs. Therefore, solutions that solve problems by anticipation are called *a priori* solutions, and solutions that correct problems after they occurred are called *a posteriori* solutions.

2.2 The resource composition model

Interoperability problems occur only when resource are put together to form a target system. The resource composition model identifies the *resources*, their *objectives* and their composition *relations*. Here, resources are defined as 'things' that can be manipulated and put in relation with others.

Relations between resources can be either *structural* (*e.g.* between a plug and a socket) or *behavioral*(*e.g.* calling a Web Service). They are 'physically' realized through an *interface* which, as a resource itself, is also part of the resource. All sorts of relations are considered, and according to our model, *communication* is a particular kind of behavioral relation.

The *objective* associated to the resource is an important concept that can be somehow related to the existence and application conditions of problems

and solutions in the decisional model. The objective of a resource states what the resource is useful for. Similarly, technical aspects can also be related to objectives of resources and systems of resources. This enables relating contexts (*e.g.* technical aspects, existential and application conditions) with resources and systems.

2.3 The systemic model

Composition of resources results in the formation of a *system*. As for each individual resource, every system subsumes an objective to achieve, which is not simply the composition of the individual objective of the composing resources.

In engineering, resources and systems are the result of a building process. The result of such building process is a system, built from an a priori *model*. Such model is most of the time established from a description of the reality (or a part of it). This description of reality also constitutes a model of a pre-existing system. From this observation, it appears that system and model relations are dual: on one hand, certain models describe existing systems with the objective of understanding the reality; while on the other hand, certain models define a new system to build. Analysis and specification models in software engineering are typically such models where one has first to understand the business and draw a model of it, and then derive another model specifying the software that will support that business. This indirect relation between the system to understand and the system to build also implies a time entailment where the model used to describe a system always comes before the model defining the new system. The relation with a priori and a posteriori solutions in the decisional model is quite straightforward. Indeed, 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 (in that case, the solution will act on models used to define the new system), or after the system has been built (in this case, the solution acts on the system itself, most often by inserting a new resource).

If models are used to define new systems, they can also be defined by other models. The *representation* of the model is important in our context as this corresponds to their materialization. Indeed, model representations are systems themselves constituted by a composition of symbols. This will be of particular importance when discussing syntax issues in interoperability.

2.4 The interoperability model

From the three fundamental models we have presented, we can now define more precisely the interoperability. As stated in the definition we adopted, interoperability is viewed as a problem, and is thus logically implemented as a subclass of the concept of problem. Such problems are related to the composition of resources, whatever the kind of relations between these resources. As resource composition is strongly connected with the concept of system, interoperability problems occur in and between systems. Since it is a problem, interoperability also bears a condition of existence, which pertains to the domain of *resource*

heterogeneity. Resource heterogeneity arises at the levels of models in the systemic part, or at the level of interfaces in the resource composition part. Models are the origin of heterogeneity when the different resources and systems to built have been made from different models. In the resource composition, the source of heterogeneity is concentrated on the interfaces. Indeed, if interfaces are homogeneous, there is no interoperability problem whatever the internal structure of the inter-related resources. An interoperability problem can obviously appear at one or both levels. Interface compatibility check when putting resources in relation will detect 'surface' apparent interoperability problems (for instance, the diameter of the screw is different from the diameter of the nut, or method signature of a library are not compatible with the calling program). Differently, if one wishes to identify more 'internal' possible incompatibilities (for instance when nut and screw are made of plastic and steel), one has to go back to the models, if they are available.

Solutions pertaining to the interoperability problem are of two kinds. A priori interoperability solutions are of the homogenization type, while a posteriori solutions are of the bridging type. As they are important concepts for interoperability, we derived two dedicated models, which we describe here.

An homogenization solution requires two basic applicability conditions to be effective: first, the modification of the system must be possible, and second, one must have a sufficient knowledge about the systems and resources to homogenize. Such knowledge is in fact contained in the models that have been used to build the systems. 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. Homogenization is an a priori solution that acts on models. Fundamentally, its objective is to transform the models defining the systems and resources that are put in relation so that the heterogeneity is removed and a new homogeneous system is re-built.Homogenization requires *transformations*, which can be either syntax transformations (for instance transforming RDF triples from their XML syntax to the N3 notation) or more intricate semantic transformations (transforming a UML class diagram in a Relational diagram involves the consideration of semantic aspects). Transformation are always made using a *unified model*, which can be of several kinds. Indeed, one can use a *unified language* to achieve homogenization (for instance, re-write all software components in a package using the same programming language, or deciding that everybody will speak English in a meeting to solve the communication behavioral relation problem), a *unified meta-model* (or ontology) to reduce semantic heterogeneity, or a *unified interface* (as *e.g.* in the universal plugs and sockets).

Homogenization introduces a series of problems. One of the most prominent one is certainly the validation or the unified model and the verification of transformations. Obviously, modifying an existing resource or system is a new source of error which may generate problems that have been solved in the original systems. In addition, performance issues due to uniqueness of resource may also arise.

When homogenization is not possible (no modification and/or no information about the legacy system is available) or when it creates more problems than it can solve, bridging is the alternative. Bridging consists in adding a new resource in the system capable of eliminating the heterogeneity. This resource is called an *adapter* and uses a *translation protocol*. These ones can act at different systemic levels and on different relation types. Indeed, a plug adapter between *e.g.* DIN and US standards acts on a structural relation at the level of the system's resources, while MOM (Message-Oriented Middleware) acts at the level of the model representation carried out by the message.

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 of the robustness, the security or the safety of the system.

3 Relations to other models related to Interoperability

To our knowledge, our attempt to define a formal model of what Interoperability is, is the first one. However some models dealing with Interoperability already exist. In the INTEROP NoE, another initiative to define more precisely interoperability conducted to the Glossary [9]. We briefly present it here. In the second part of this section, we discuss some of these models, related in a study conducted for the US Department of Defence (DoD) in 2004 [8], and compare them to our proposition. Finally, we suggest how they could be used with the OoI and show that they should be describable using the OoI. The third part of this section presents the Model Morphisms Ontology (MoMo) which deals also with interoperability but in the domain of models and might be used with our ontology as a solution for interoperability problems between models.

3.1 The Glossary

Work on ontologies of interoperability is still very scarce. However, one effort in this area has been undertaken within the INTEROP NoE, where a task group has developed a glossary for interoperability concepts [9]. In contrast to the OoI, the starting point for this work has been the actual use of interoperability terminology among researchers and practitioners. The Glossary aims at recording terms frequently used in the area and structuring them in a hierarchy. Thus, the glossary does not provide complex relationships among interoperability concepts but only a taxonomy. It is therefore provided as a textual document targeting to provide definitions that can be read by human beings. In opposition to this, the OoI does not target to provide a readable definition but a formal ontology that may be interpreted by *e.g.* a special software.

However, it can still be useful to compare the Glossary efforts with the OoI. A preliminary comparison that has been undertaken shows many interrelationships. One finding is that the OoI contains more specialized concepts than those

of the Glossary which suggests further refinements of the Glossary. The Glossary on the other hand can enrich the OoI by supplying established terms in the interoperability domain.

3.2 Models for Interoperability

There exist a few models of Interoperability. Among them, a widely recognized one seems to be the LISI (Level of Information Systems Interoperability) model [4], which as been used in 2003 to adapt the reference model used by NATO: NC3TA [8]-pp8. Recently another model called Systems Of Systems Interoperability (SOSI) [8] has been proposed as a result of a US Department Of Defence project. The final report summarizes some other models, among which LISI, dealing with interoperability at different levels: while LISI deals with technical level only, other models address other aspects like organizational, environmental, or operational ones. SOSI in itself, proposes a model based on three types of interoperability, namely: operational (between systems), constructive (between organizations handling the system build and maintenance) and programmatic (between different program offices). A summary of models of interest is given in figure 2.

	LISI **technical interoperability**	**OIM** **organizational interoperability**	**LCIM** **conceptual interoperability**
Level 4:	Enterprise: universal environment (Interactive manipulation, shared data and applications)	Unified	Harmonized data
Level 3:	Domain: integrated environment (Shared data, "separate" applications)	Integrated	Aligned dynamic data
Level 2:	Functional: distributed environment (Minimal common functions, separate data and applications)	Collaborative	Aligned static data
Level 1:	Connected : pear to pear environment (Electronic connection, separate data and applications)	Ad hoc	Documented data
Level 0:	Isolated: manual environment (Non connected systems)	Independent	System specific data

Fig. 2. Interoperability maturity models. The Levels of Information System Interoperability (LISI) model defines technical aspects; The Organizational Interoperability Maturity Model (OIM) [2] is an equivalent from an organizational point of view; and the Levels of Conceptual Interoperability Model (LCIM) [13], at the conceptual level, focuses on data and how they are made accessible through documentations and interfaces. For more details, refer to the SOSI report [8] or to the referenced papers.

From this report and previous models, it is clear that interoperability is not specific to the technical layer. Human-related layers play also a role and a part of them like e.g. business needs are even at the origin of interoperability problem [10]. As for the moment we deal mainly with technical aspects, though using a conceptual vision and pursuing a goal of generality, we will not push the discussion further. Concerning these models, a number of interesting facts can be derived:

– Interoperability is seen as a requirement a system needs to fulfill. It is not seen primarily as a problem, though the objective of interoperability induces directly problems.
– All the models specify different maturity levels, defining which degree of interoperability an existing system may have, or what kind of relationship between systems should exist to make a system of systems reach a given degree. In this way, they are models *for* interoperability rather than models *of* it.
– As with the older NATO NC3TA model, Interoperability seems to be reduced to data or information transmission or sharing between systems. Interoperability is then always linked to communication and the models do not take into account the need for structural contact, which we define as an interoperability problem and relates simply to interface compatibility and not necessarily communication (see the electrical plug example in [11]).

While we try to achieve a quasi-formal description of what Interoperability is, so that it can be used to find and solve interoperability problems in systems, the maturity models we cited provide descriptions of how systems should be structured to get a given level of interoperability between their components. The goals are different, and are in fact complementary. From the point of view of our ontology, maturity models could be associated to specific sets of solutions to identified interoperability problems. Once all (or a sufficient number of) the solutions of such a set are used in a system, this one would then be stamped as being interoperability level-X compliant according to a chosen maturity model. With LISI, this could be achieved using the interoperability profiles defined with the PAID method [4], as solutions in the form of Procedures, Applications, Infrastructure, and Data related recommendations or requirements.

As the description of each level defines solutions pertaining to interoperability problems, it should be formalized using the OoI. We illustrate this with the LISI model, on its level 0. LISI level 0 describes interoperability between isolated systems. Each system can be considered as being a *resource*, part of a bigger system. These resources need to transmit data. At design time, *interoperability problems* are solved *a priori*, using adequate *interfaces* and *bridging solutions*: e.g. a disk is a bridge, linked to the physical disk controller of computer systems. But this is only the visible part of the iceberg. Pushing further, as soon as we use the disk as solution, we enable data transmission between systems, but other *induced interoperability problems* need also to be solved: file system compatibility, vocabulary used in the data, the semantics, etc. Hence it can easily be seen that we can follow the reasoning described in the OoI to express the simplest level of the LISI maturity model. Other levels can be expressed as well, however the more complex the system, the more layers of problems-solutions and induced problems there will be. This complexity, which is inherent to interoperability, induces logically the need to have a more complete as possible formalization if we want to automate the finding and resolution of interoperability problems.

3.3 The Model Morphisms ontology (MoMo)

Modeling is used in almost all areas of (business) information systems today. Models help to keep an overview about complex scenarios and they abstract from the reality. Today, there are plenty of formats and approaches for performing modelling within all major areas. For example, in the database design, Entity-Relationship models (ER) are very popular. When developing software architectures, UML has become a major standard for modelling.

One can distinguish between different types of models as presented in [3]: diagrammatic (e.g. EPC, UML, E-R, CG), logic-based (e.g. F-Logic, Situation Calc, FOL, DL, Prolog, Datalog), functional (e.g. KIF, OntoLingua, PIF, PSL), XML-based (e.g. XMLS, OWL, XMI), and algebraic(e.g. Z++, Obj, Pi-calc). Most of these models can be applied using tools that support their usage such as, e.g. Together[5]. However, the main problem in this area is that most models are incompatible since they are developed independently. Furthermore, many of them are overlapping or could be used for similar tasks. In order to address those problems, the task group MoMo of the INTEROP NoE conducts researches aiming at providing a toolbox that helps to select a modelling language for specific needs. The main research area of MoMo is the connection of multiple models by using morphisms: especially model transformation, model matching/mapping, and model merging.

In order to achieve its goal, MoMo starts with the creation of a so called Model Morphisms (MoMo) ontology that represents models and their relationships. This ontology aims at helping a user to select a modelling language according to his specific needs. This makes it necessary to analyse the relationships between different modelling languages and hence, it is necessary to analyse their interoperability issues. The link between MoMo and the OoI, proposed in the next section consists in the application of MoMo as one specific use case that can be described by the ontology of interoperability. We created instances in our ontology model that expresses the problem of selecting specific techniques of the model morphism domain. Afterward we added the MoMo framework as a possible solution to this task.

4 Illustrative use case

This section illustrates how the OoI could be used for representing the interoperability problems covered by the Model Morphisms ontology. The MoMo ontology itself tries to solve a part of the interoperability problem: interoperability between models. Hence, the OoI can be used to model an interoperability problem between two modelling standards and the MoMo can be linked with it as one possible solution to this problem. In order to realize this, we started from two modelling standards: EPC [7] and BPEL [1], for which we wanted to find mappings to solve interoperability issues.

[5] http://www.borland.com/de/products/together

The Event-driven Process Chains (EPC) model has been developed within the ARIS framework [12] in order to model, analyse and redesign business processes. It has been developed from a graphical view point allowing human beings to easily deal with business process chains up to a certain degree. The Business Process Execution Language (BPEL) is closely related to Web Services and allows to build business processes by defining a number of Web Service orchestration concepts (see e.g. [1]). BPEL and EPC are closely related. Compared to EPC, BPEL is more technical oriented and was not necessarily developed for graphical modelling. Of course it might be possible to convert business processes from e.g. EPC to BPEL but there are of course a number of interoperability problems. For example, both standards use a different syntax for expressing business processes and also a completely different semantics.

This example shows one specific scenario for two technologies that could be used to solve a specific problem. Both concepts (BPEL and EPC) are used in the same domain but they differ in their functionality and in their capabilities. Transforming a process from one standard to the other is certainly not an easy task and needs to consider various interoperability problems. This is where the OoI comes in. It can be used to describe the interoperability problem in this area and it can also model possible solutions for the interoperability problems. For example, if we think of the problem of selecting one of those standards (BPEL or EPC) for a specific tasks then we could for example interpret the MoMo framework, described in the las section, as a possible solution. In order to proof if the OoI can be used in such a scenario, we have added both EPC and BPEL standards as separate instances of the *momo:modelling language* concept. Furthermore, we created an instance of the concept *ooi:solution*, which represented the MoMo framework. Afterward, we finished the case study by adding connections between EPC and MoMo as well as between BPEL and MoMo. The outcome is a description of the interoperability problem between EPC and BPEL in the OoI. Moreover, not only the problem was modeled but also one possible solution, which is the MoMo framework.

5 Conclusion and perspectives

We have presented here our attempt to provide a formal model for interoperability in the form of an OWL ontology; namely the Ontology of Interoperability (OoI). The approach and concepts used have already be explained in previous papers [10], [11], where the model was presented in the form of UML representations. Here, we begun a comparison with existing models of maturity for interoperability and illustrated possible connections with our ontology. In particular, we have shown that the ontology might be used to describe formally interoperability problems in the different levels of interoperability defined by maturity models. We suggested also that these later could be used for defining set of solutions to interoperability problems specific to a particular level of interoperability maturity, and thus instantiated using the OoI. The proposed OoI ontology has shown a comfortable way of expressing interoperability problems.

Furthermore, we have given a practical example that shows how the problems that arise within the MoMo framework can be represented with the OoI.

In a next step we will use our ontology to model some real-world interoperability problems. This will be performed by describing typical scenarios and giving a representation in the ontology of interoperability. We also plan to investigate further maturity models and trying to formally link one of them to our ontology. Also as we found mainly US initiatives, we would like to investigate European equivalents. As for the moment, the OoI while being a conceptual model, deals mainly with technical aspects, it would also be interesting to model the multiple facets of interoperability that are rather linked to human aspects.

References

1. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana, *Business process execution language for web services, specification version 1.1*, Tech. report, 2003.
2. Thea Clark and Richard Jones, *Organisational interoperability maturity model for c2*, in Proc. of the 1999 Command and Control Research and Technology Symposium (U.S. Naval War College, Newport, RI, Whashington D.C.), June 29-July 1 1999.
3. Fluvio D'Antonio, *Momo - task tg3.2: Roadmap*, presentation, INTEROP Workshop, 2005.
4. C4ISR Interoperability Working Group, *Levels of information systems interoperability (lisi)*, Tech. report, US Department of Defense, Washington, DC, 1998.
5. IEEE, *Ieee standard computer dictionary: A compilation of ieee standard computer glossaries*, Institute of Electrical and Electronics Engineers, 1990.
6. INTEROP, *European network of excellence*, 2004.
7. Gerhard Keller, Markus Nuttgens, and August-Wilhelm Scheer, *Semantische prozessmodellierung auf der grundlage ereignisgesteuerter prozessketten (epk)*, Publication of the institute of business information systems, Saarbrucken, Germany, vol. 089, 1991.
8. Edwin Morris, Linda Levine, Craig Meyers, Pat Place, and Dan Plakosh, *System of systems interoperability (sosi): Final report*, Tech. report, Carnegie Mellon Software Engineering Institute, Pittsburgh, USA, April 2004.
9. Raul Poler, Jose V. Toms, and Paola Velardi, *Interoperability glossary*, Tech. report, Interop NoE Deliverable D10.1, http://interop-noe.org/deliv/d10.1M18, 2004.
10. Vincent Rosener, Thibaud Latour, and Eric Dubois, *A model-based ontology of the software interoperability problems: preliminary results*, in proc. of CAISE04 workshops, EMOI 04, vol. 3, 2004, pp. 241–252.
11. Vincent Rosener, Yannick Naudet, and Thibaud Latour, *A model proposal of the interoperability problem*, in proc. of CAISE05 workshops, EMOI 05, vol. 2, 2005, pp. 395–400.
12. August Wilhelm Scheer, *Business process engineering: Reference models for industrial enterprises*, Springer-Verlag Telos, 1994.
13. Andreas Tolk and James A. Muguira, *The levels of conceptual interoperability model*, 2003 Fall Simulation Interoperability Workshop (Orlando, Florida, U.S.A.), Sept. 2003.