Undefined 1 (2009) 1–5
IOS Press

Future Internet Ontologies: The NOVI Experience

Wibisono Adianto [a,*], Cees de Laat [a] and Paola Grosso [a]
[a] *System Network and Engineering, Informatics Institute, University of Amsterdam E-mail: {a.wibisono,delaat,p.grosso}@uva.nl*

Abstract.
Semantic web technologies have been applied in the last years to an area where they were rarely used before: the Future Internet (FI) research field. In this context OWL ontologies are used to describe resources and services provided by FI platforms.

Few distinguishing factors set application of Semantic Web in this area apart. First, the ontologies and the corresponding data models are integral part of the software development process. Second, the definition of (new) operations and services in these infrastructures forces the FI ontologies to continually evolve. These features make the mapping of the information model into the data model a crucial aspect, furthermore changes in the information model need to be automatically propagated.

In this paper we present the ontologies developed in the NOVI EU-funded project, their use to provide the required functionalities, and we will in particular focus on the benefits and drawbacks of using direct and indirect mapping, and discuss the lessons learned from this experience, and how they can benefit the FI community at large.

1. Future Internet and Semantic Web

Future Internet (FI) research defines novel architecture for the internet; in recent years lots of effort in this area has tried to implement a core and essential feature, namely *federation* of infrastructures.

Several projects attempted and are attempting to achieve federation by adopting semantic web techniques, *ontologies*.

Federation of heterogeneous internet resources belonging to different providers is in fact crucial in these new internet models. The motivation for this is clear: services will be more easily offered by using a plethora of devices, and only in very few cases a single domain will have all the types of resources needed by an end-user. All of these envisioned FI federations share several crucial features: common authorisation policies, coordinated monitoring of available resources, and intelligent selection of available computing and networking resources.

The adoption of semantic web ontologies as underlying information model for resources and services has also a clear motivation: ontologies allow deduction of service and infrastructure behaviours. Assuming that all resources in an FI platform are modelled in an ontology, assuming that there are ontologies for policy and monitoring data, we can intelligently deduce which resources are available and which services can be provided. The application of semantic web technologies for integrating heterogeneous data enables advance analysis.

An example of this is the recently ended NOVI project, sponsored by the EU under the FP7 program. We participated in NOVI where we used OWL ontologies as formalization of the information models and we developed the corresponding data models to enable the communication among the various components in NOVI software architecture. The NOVI information model describes resources at a conceptual level, including all the components required to support the operation of the NOVI software. The data model describes implementation details based on representation of concepts and their relations provided by the in-

*Corresponding author. E-mail: a.wibisono@uva.nl

0000-0000/09/$00.00 © 2009 – IOS Press and the authors. All rights reserved

formation model. The NOVI ontologies captured the requirements at the design and analysis phase of the services that were being built.

In Section 3 and Section 4 we report on the ontologies that have been developed in NOVI and on their use in the software architecture.

We derived useful lessons from our experience in NOVI; our work proved us that the advantages of semantic modelling are counterweighted by several challenges:

1. the mapping of concepts in the ontology to the object oriented structures and processes used in the projects is an important decision, which requires careful consideration depending on the initial state of the software.
2. there is mismatch in timelines between the definitions of the information model made during design and requirement phase and the development of services and component at the implementation phase; this situation creates challenges in adaptation of service development to changes in information model. The chosen mapping will influence the ease of making changes.

We will elaborate on these problems in Section 5, Section 6 and will provide our answers and recommendation in Section 7 and Section 8.

Several other initiatives, such as the GENI [2] and the Fed4FIRE [1] projects, are following suit and introducing ontologies as their base information model. The conclusions we provide in this article are of importance for them and for all other FI projects intending to adopt semantic web modelling. In particular the two problems we identified will require particular attention and they constitute a valuable legacy of the NOVI work.

2. Existing models

The NOVI project participants identified several features that needed to be supported by the NOVI ontologies: support for virtualization concepts, context-awareness, support for monitoring and measurement concepts, support for management policies. The obvious question was to determine if existing models in other Future Internet related projects could already provide (some of) these features. A thorough review of existing models focused on the following models: CIM[6], DEN-NG[13], MOMENT[3], NDL[14].

The existing model's evaluation lead NOVI to conclude that only minimal part of some of the existing ontologies could be reused; in fact:

- CIM models infrastructure resources using UML. It did not provide support for semantic and context awareness, neither for monitoring and measurements concepts and for this reason it was not further considered.
- DEN-NG and MOMENT were complementary works with respect to NOVI, since the first is focusing on policy management while the latter if focused on monitoring resources. DEN-NG started with UML based approach which then later realized to be lacking in ability to represent behavioral semantics. They complemented the original approach with the use of ontology side by side with the original ontology.
- The main innovation of the MOMENT was the use of a measurement-specific ontology, allowing semantic representation and retrieval of measurement and monitoring information, as well as providing the flexibility of a service oriented architecture for future Internet applications.
- NDL did not support virtualization concepts and monitoring/measurement data, however, its support for describing network topologies and device configuration was ultimately used as a building block for the NOVI information model.

More information on the relation of the emerging NOVI model and the existing models can be found in [16] and [17].

3. The NOVI ontologies

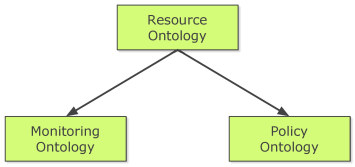

Fig. 1. The three NOVI ontologies: the resource ontology provides the basic concepts that are used in the monitoring ontologies and in the policy ontology.

The OWL ontology used in the NOVI projects consists of three modular components [15], which are depicted in Fig. 1. There is resource ontology, and two ontologies that are based on it: the monitoring ontology and a policy ontology.

3.1. NOVI Resource Ontology

The main elements of the NOVI Resource information model can be seen in Fig. 2. The classes Node, Network Element, Node Component and Service are all subclasses of the generic Resource object.

- Node is used to represent physical nodes, while a subclass VirtualNode is used to represent virtualized nodes;
- Network Element is an abstract class with three specific subclasses: Interface, Link and Path;
- NodeComponent is also an abstract class that describes essential components of nodes that are of interest to the user, such as CPU, Memory, Storage, SwitchingMatrix and LoginComponent;
- Service class allows the user to express the service level desired and to decouple the desired service from the actual physical implementation.

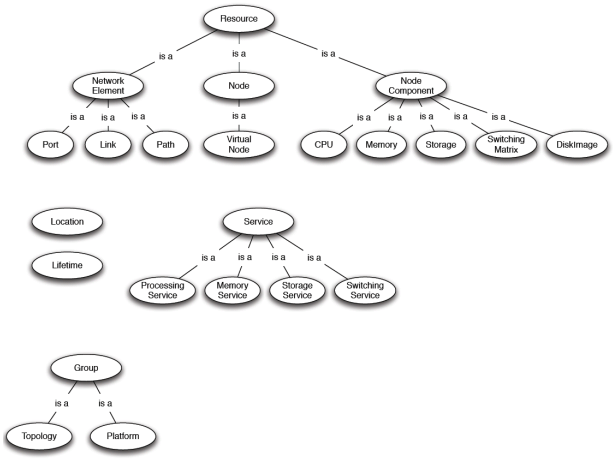

Fig. 2. The NOVI Resource Ontology

Besides the resource description classes there are also other classes to describe properties of resources:

- Location provides a way to describe location of resources.
- Lifetime describes the time dimension of other objects, such as reservations, or availability of nodes.
- Group is an abstract class to describe groups of resources which provides support for essential concepts within the NOVI federation, namely the Platform and Topology subclasses. A Platform describes a particular testbed in the NOVI federation. Resources can be linked to the class to denote membership of that platform, and it can provide pointers to other information such as the management service of that platform. A Topology can define a group of resources that a user requests, or the implementation of a usersâĂŹ request.

3.2. NOVI Monitoring Ontology

For its operation the Monitoring Service present in the NOVI software architecture requires formal descriptions of different groups of concepts, their relationships and properties:

1. Resources: to describe the resources to be monitored. These concepts are defined in the resource model as described in the previous section;
2. Monitoring Tools: to describe monitoring tools and their parametrization. The main goal of this part of the model is to enable the Monitoring Service to translate incoming monitoring queries to platform-specific commands;
3. Monitoring Data: to describe the monitored properties. This part of the model helps the unified, platform-independent and unit-aware handling of monitoring results;
4. Data manipulation: to describe the flow of transformation steps, which can be applied to the data, such as re-sampling and aggregation.

The Monitoring ontology is also built in a modular manner and provides support for all the above. Fig.3 shows the classes that are defined in the feature part of the model.

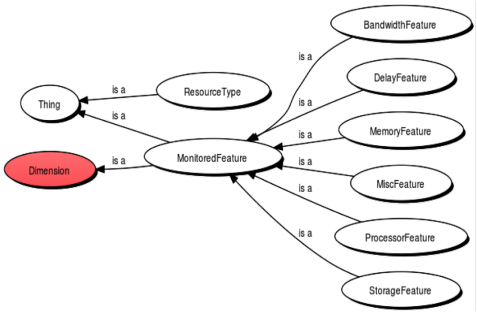

Fig. 3. The Feature component of the NOVI Monitoring Ontology

3.3. NOVI Policy ontology

The NOVI policy ontology provides the conceptual support to the operation of the Policy Service.

The Policy service provides support for authorization policies that specify which subjects may access virtual resources within the federation; it enables event-condition-action policies that enforce control and management actions upon certain events. Figure 4 shows the class hierarchy in this ontology.

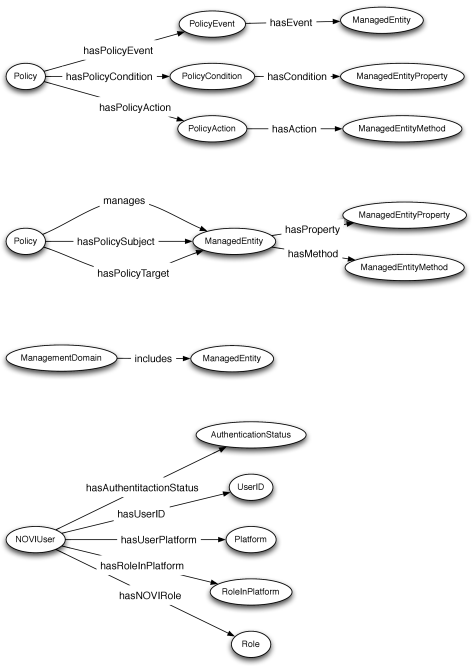

Fig. 4. The NOVI Policy Ontology

4. The NOVI architecture and its use of the ontologies

The NOVI ontologies provide support for the operation of the software components in the NOVI architecture. Figure 5 shows a federation consisting of a testbed A and a testbed B.

Each platform has its local set of NOVI software instances. The ultimate goal of NOVI is to allow the creation and management of the virtual layer *slices* by finding, gathering and configuring the proper services on the underlying physical resources. On each platform the various services are only responsible for the local resources; in case of cross-platform resource discovery or monitoring, the local instances communicate with each other via their neighbour services. All communication between components in the NOVI layer is mediated by the NOVI ontology.

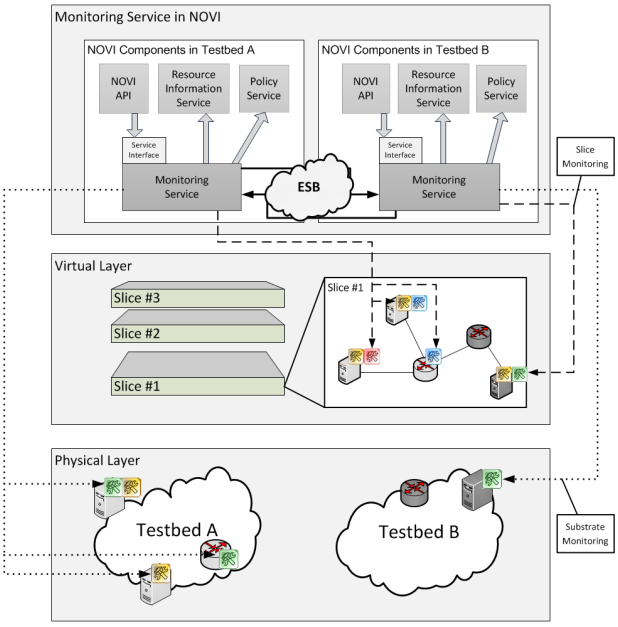

Fig. 5. The NOVI Architecture supporting the federation of two platforms: testbed A and testbed B. *Source: [7]*

The NOVI layer defines the following services: the IRM (Intelligent Resource Mapping), the PS (Policy Service), the RIS (Resource Information Service), the MS (Monitoring Service) and the API. The best way to understand the relations and the operations of these components is to follow a user request for a slice as it travels through the system.

A user will submit requests for standalone virtual resources, topologies of virtual resources and specific services regarding virtual resources/topologies. Requests can be of two types: bound and unbound. A bound request contains the mapping between the virtual and physical resources that the users wishes to have; an unbound request will leave the NOVI system free to perform mapping within the specified constraints. To compose this request NOVI users rely on the graphical editor that has been adopted in the project[18]. The NOVI editor helps the platform user to assemble and identify the resources they need without having to write the OWL topology.

Figure 6 shows an unbound request for one virtual node with storage 10GB storage and 4-core CPU with a 2 GHz processor. In Listing 1 we highlight part of the request (expressed in Turtle) corresponding to the definition of the Virtual Node *vnode1*, the CPU and the storage characteristics.

Listing 1: Unbound request

```
:requestTopology NS5:type :Topology,
     owl:NamedIndividual ;
     :autoUpdateOnfailure "false"^^xsd:boolean ;
     :contains :vNode1 .

###  http://fp7-novi.eu/im.owl#vNode1

:vNode1 NS5:type :VirtualNode ,
                owl:NamedIndividual ;
     :hasVendor ""^^xsd:string ;
     :hasVirtualizationEnvironment ""^^xsd:string ;
     :hrn ""^^xsd:string ;
     :virtualRole ""^^xsd:string ;
     :hostname ""^^xsd:string ;
     :hasOS ""^^xsd:string ;
     :diskImage "0"^^xsd:anyURI ;
     :hasAvailableLogicalRouters "0"^^xsd:int ;
     :hasLogicalRouters "0"^^xsd:int ;
     :exclusive "false"^^xsd:boolean ;
     :hardwareType "x86_64"^^xsd:string ;
     :hasComponent :vn1-cpu ,
                   :vn1-sto .

###  http://fp7-novi.eu/im.owl#vn1-cpu

:vn1-cpu NS5:type :CPU ,
     owl:NamedIndividual ;
     :hasAvailableCores 0 ;
     :hasCPUSpeed "2.0"^^xsd:float ;
     :hasCores 4 ;
     :exclusive "false"^^xsd:boolean .

###  http://fp7-novi.eu/im.owl#vn1-sto

:vn1-sto NS5:type :Storage ,
     owl:NamedIndividual ;
     :hasAvailableStorageSize "0.0"^^xsd:float ;
     :hasStorageSize "10.0"^^xsd:float ;
     :exclusive "false"^^xsd:boolean .
```

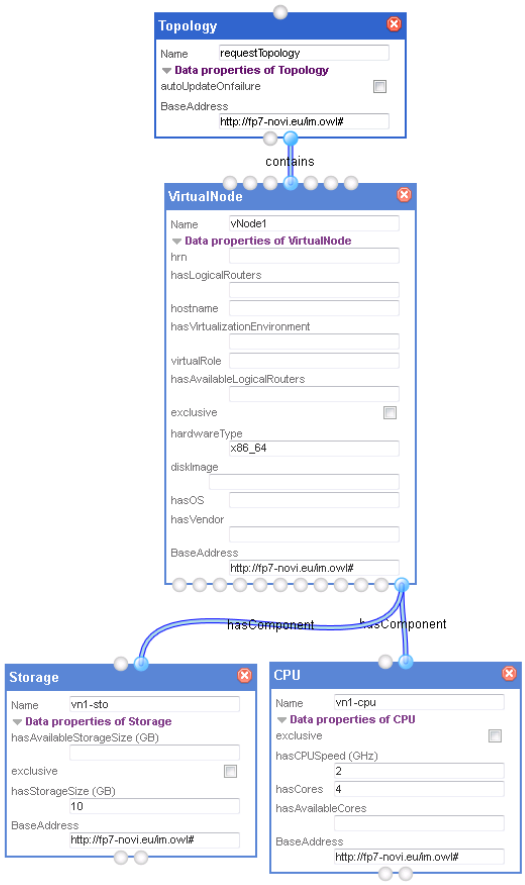

Fig. 6. Unbound request for one Virtual node with cpu/storage requirement

The NOVI API is the users main entry point to the NOVI layer; it delivers this request to the appropriate NOVI services such as the IRM or Policy Manager service. The Policy Service validates the credential of the user, while the IRM service will ultimately embed the user requests for virtual topologies / resources (Virtual Networks - VNs) to the federated physical substrate network. To perform the embedding the IRM first gathers information from the RIS/ and the MS regarding available resources.

The RIS acts as a single point of contact for other NOVI services to acquire information about resources' status. One of the functionalities of the RIS is to register and discover resources. Resource discovery encompasses locating and retrieving information across the federated virtualized substrate network. After retrieving this information, the RIS must provide accurate current status of resources to other services.

The MS collects information about specific resources and metrics. The monitoring can be performed from slices (virtualized resources) or from hosts (physical devices). In addition, it is possible to obtain passive monitoring information from resources or from repositories, and active monitoring information, where it is needed for the operation of the network elements.

Only resources that satisfy the requirements imposed by the VN request are selected. If resources are required across testbeds, then the IRM will solve the so called inter-domain Virtual Network Embedding problem[10].

5. Model mapping methodologies

An important lessons learned during the NOVI project has been the need to decide on the proper mapping techniques. In this section we will discuss different approaches on how to use ontologies and semantic web technology to develop software components for Future Internet platforms. In NOVI this approach con-

siders the conceptual relationships captured by the information model formalized as ontology described in section 3, to derive an object model that can support the operations and behaviour of the NOVI software services described in section 4.

Using ontology to drive software development is seen as a natural next step in the Model Driven Development [9]. In [4] domain models are used as a bridge between underlying relational persistence model and the object model. It is also possible to develop a component that fully corresponds to an ontology. Different domains and ontologies can share components from repositories. This is obviously very interesting for Future Internet project who faces similar challanges with regard to platform federations and network embedding and are keen to reuse software when available.

Ontologies should be, in a sense, a basis for designing and developing interoperable software components in practice, since they can precisely define the semantics of components and their parts, as well as the types of relations and communication between software [5].

Generalizing from our experiences, we identify two main mapping approaches:

- **direct object model mapping**, where object model is derived directly from the underlying information model, maintaining exact conceptual relationship. These model is then used between components using this type of mapping and within the same component. We call the components using this approach *direct-mapping components*;
- **indirect object model mapping**, where object model are indirectly derived, and taking more into consideration on how the conceptual relationship can be adapted to suite existing software components/software services. This approach performs the necessary mapping only to extract the relevant information from the information model into the internal representation of a possibly existing software services. Components adopting this type of approach will be called *indirect components*.

In Fig. 7 we illustrate the difference between these two types of mapping. In the direct approach, the object model in the software artifact corresponds directly to the conceptual structure of underlying domain ontology; the software's internal representation of the objects is structurally similar to the original domain ontology, *i.e.* the red, blue and green classes have the same relation in both the ontology and the artifact.

Meanwhile in the indirect approach, the software artifact has its own internal representation, and the relations between classes and objects can be different than in the original ontology. Objects and classes will be mapped into the structure identified by the domain ontology only when the information is consumed outside the system.

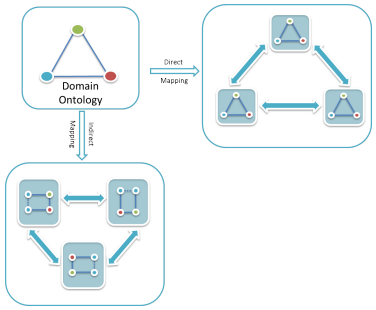

Fig. 7. Direct and indirect mapping of ontology concepts: we see the relation between the three objects in the ontology *(upper left)*, the direct representation *(upper right)* and the indirect representation *(lower left)*

5.1. Direct model mapping

In this approach, concepts in an ontologies are automatically converted into object model counterparts. This approach is exemplified for example through automated generation of Java objects that represents the underlying ontology. The motivation for choosing this mapping is to isolate the object oriented programmer from knowledge of ontologies, or having to be familiar with knowledge engineering methods. One of the advantage of direct object mapping is that it well suits domain aware application, where the mapping make sense.

Example of a direct mapping can be seen in figure 8. Here three concepts in the ontology, namely the Node, Interface and IP address, are related to each other by predefined properties: hasInterface and hasIPAddress. The mapping creates three Java objects with the same name and with the same relations.

5.2. Indirect model mapping

Object structures in the indirect mapping can be different than the one in the original ontology. We can see this exemplified in Figure 9. Here the original classes Node, Interface and IPAddress are mapped into a single Java object. The relation/properties between them

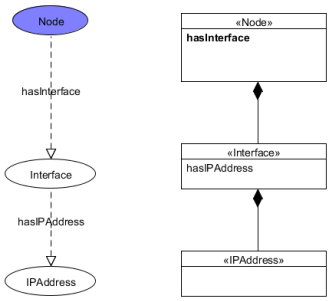

Fig. 8. Direct mapping of ontologies: the relation between the concepts Node, Interface and IP address in the ontology *(left)* are mapped to Java objects in the software artifact *(right)*

are also not necessarily conserved. The mapping into the underlying ontology representation is still needed to communicate with external software components using the same basic information model.

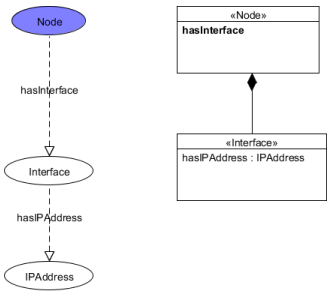

Fig. 9. Indirect mapping of ontologies: the relation between the concepts Node, Interface and IP address in the ontology *(left)* are mapped into a single Java object in the software artifact without maintaining the original structure *(right)*

Semantic web technology libraries such as Jena or Sesame provide a generic object model mapping that can be used by developers. In this mapping, the java objects that developer have to work with will not be domain specific, but will be only primitives from the underlying ontologycal terminologies. Developer will have to use for example OntologyClass, or NamedIndividual without dealing directly with domain specific objects.

5.3. Hybrid model mapping

There is a third possible way to perform mapping: an hybrid approach. As proposed in [11], the hybrid mapping approach combines the direct and indirect mapping. There are some core sections where the object model corresponds directly to the information model, while for some other sections the object model is indirectly derived from the underlying information model. The reason for keeping some section of the object model indirectly mapped is scalability: for domains where the underlying original ontology that needs to be represented is very large it is no longer feasible and efficient to perform direct mapping.

6. Mapping in NOVI

We applied the two approaches we described in Section 5 in the NOVI project. Direct mapping of core NOVI Information Model was used for all software components, except for the Monitoring Service which adopted the indirect mapping.

6.1. NOVI direct mapping with Alibaba

The NOVI Information model is directly mapped using the Alibaba tools[8]. Alibaba is intended to combine the flexibility and adaptivity of RDF with an powerful Object Oriented programming model. Among several features provided by this library, we are focusing on the Object Repository which allows us to perform direct mapping of OWL/RDF into objects model.

AliBaba's object repository provides programmers with increased expressivity and a simplified subject-oriented programming environment. The Object Repository is an extension to the Sesame RDF Repository that allows an RDF store to function as an object store. It maps Java objects to and from RDF resources and OWL classes to Java classes in a non-intrusive manner that enables developers to work with resources stored in an RDF Repository as objects. The Object Repository may also optionally be configured with a BLOB store, to store information-resources.

In an ontology concepts are a hierarchical model of resource classes, that include a description of supported operations on a type, including syntax and semantics. A concept defines the properties and methods available to objects retrieved from the store. Alibaba allows to manually perform the *concept mapping* between existing Java classes and the concepts in an ontology; it also allows to automatically perform *concept generation* of Java classes and interfaces based on existing ontology. Mapping and generation have the following characteristics:

1. **Concept Mapping**
   With Alibaba concepts are mapped to Java classes or interfaces annotated using IRI (Internation-

alized Resource Identifier). The IRI can be assigned explicitly during definition of classes, or at runtime. RDF objects retrieved from the store implement all the concepts that map to one of the URI/IRI rdf:type values of the RDF resource. Any Java field, getter method, or setter method, on a concept, that includes an @Iri annotation will be mapped to an RDF property using the given predicate.

2. **Concept Generation**
The process of mapping concepts in the original OWL/RDF file to the Java object classes can be performed in two manners: either manually by annotating the classes that are going to be stored in the object repository, or by using owl compiler provided by Alibaba. In our implementation in NOVI we use the existing OWL compiler script as a base and perform our own customization, where we did a small modification on the way packages are generated.

6.2. Indirect mapping of NOVI Monitoring IM

In this mapping we used the ontology to allow the Monitoring services to handle heterogeneous federation platform. Part of the monitoring software building blocks had been developed prior to the NOVI project and existed before the NOVI Monitoring ontology was fully defined. These services already had their own internal representation of concepts with relationship.

Indirect mapping used an intermediate semantic storage between the monitoring services and the other service components. All information that is going to be consumed by the monitoring services components is first loaded into a temporary (in memory) semantic store using *RDFLib* python library.

Transformation into internal representation is then performed through SPARQL Query into this temporary semantic store. The results of these query is converted into internal python object/representation.

6.3. Communication between NOVI components

In Figure 10 we illustrate how components using direct mapping and indirect mapping communicate. RIS and IRM are both direct mapping components and communicate directly using the common object model that is derived from NOVI information model. The communication between RIS and MS is instead indirect, therefore it is intermediated with an RDF store as explained in the previous subsection. The RIS

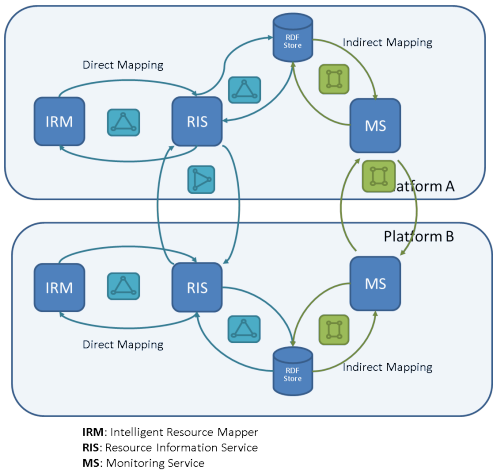

Fig. 10. Communication between NOVI components

communicates between federated platforms through direct mapping communication. An intermediate Alibaba triple store between platforms can also be used, in case the available communication protocol does not allow to direct transfers of object among platform.

6.4. Mapping issues in NOVI

During the course of the NOVI project we encountered a number of problems related to the mapping choices made. We will introduce a couple of these situations and in the next section we will generalize our findings from NOVI to any Future Internet platform considering mapping.

1. At a certain point in the project the Path concept captured in the base ontology needed to be remodelled. Originally all Path objects were represented as unidirectional collections of Links. During the development of the NOVI Services we found out that we need to have bidirectional representation of the Path, for convenience since these are the types of Path that is mostly needed in our testbeds. This change introduced new concepts, *i.e* the Bidirectional Path and the Bidirectional Links. The creation of Bidirectional Paths and Links reflected in new triples in the RDF store. This change had different effects depending on the mapping adopted:
   - the Object model from the direct mapping needed to be regenerated, and all existing code which used the unidirectional model of Path had to be adapted to this newly introduced

concept. Existing pairs of unidirectional Path objects which were already stored in the system required to create a corresponding Bidirectional link.
- in the indirect approach this changed how the Path objects coming from the triple store were mapped into the internal representation (of the monitoring service components). The changes required are only in the interpretation of the query results related to the Path.

This in a concrete example of what we will call generically *ontology changes scenario* in Sec. 7.

2. In the project we gave attention to scalability issues. In particular we identified those aspects of scalability that are crucial for adding new resources and testbeds. Fig. 11 and Fig. 12 show respectively the response time of the NOVI system for the creation of a slice when the number of resources in the testbed and the number of nodes in the request increase. These results provide information on the behaviour of the whole system; the effect of the mapping choice made by the components on the scalability cannot be easily disentangled.

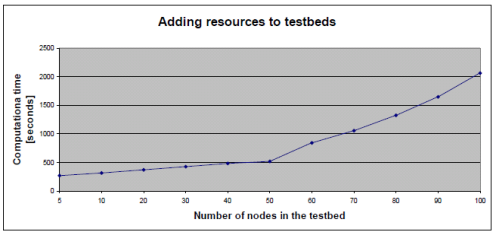

Fig. 11. Scalability result: time needed for slice creation versus number of nodes in the testbed *Source: [7]*

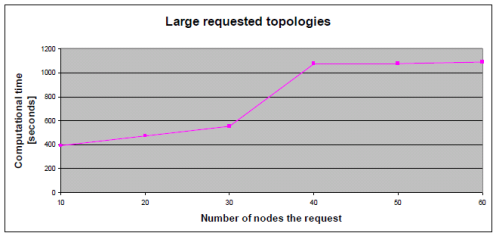

Fig. 12. Topology scalability result: time needed for slice creation versus number of nodes in the request *Source: [7]*

This in an example of *software scalability scenario* that involves both direct and indirect mapping components.

7. Evaluation

Thanks to the experience gained in NOVI we identify a set of challenges and propose solutions of interest to any other Future Internet project adopting ontologies.

We identify three major challenges all projects face during the development and further maintenance of their software. Here we are going to focus on how we are coping with these issues, with respect to the mapping approaches that we have outlined in the previous sections. We consider:

- Ontology Changes, *i.e.* changes in the underlying ontology maintaining software components behaviour;
- Software Scalability, *i.e.* how the software components behave when the ontology grows in size;
- Domain Changes, *i.e.* domain changes where we are trying to generically apply the same software services to different domains;

. For each one we indicate the implications of the adopted mapping strategy, and we try to determine if one approach is preferable to the other.

7.1. Ontology Changes Adaptation

For changes in underlying ontology where new relationships or concepts are introduced or existing relationships are changed, direct mapping and indirect mapping will need similar adaptation at the level of object model. The newly introduced concepts and relationships would need to be included in the automatic generation of the object model for direct mapping, meanwhile for indirect mapping a new conversion to internal structure of the indirect mapping would be necessary.

There will be a different implications at the level of software components and services using the generated object model. Direct software component would need to adapt to the newly introduced components and structure, while indirect component will require less modification as long as all the conversion to internal representation in the indirect mapping approach already cover the new changes.

In face of this type of changes, to determine whether direct mapping or indirect mapping is easier depends on the existing internal representation of concepts and relationships existing in the indirect mapping. If there are no new concepts introduced, but only the ontology structure changes, then it is simpler to adapt for

a project using indirect mapping. In essence a Future Internet project will cope better with changes of this type if there is a stable interface between the software and the ontology so as to allow each to evolve independently [12].

7.2. Software Scalability

The direct and indirect mapping approaches differ in how the developed software scales with respect to the underlying ontology. Indirect mapping components and applications are loosely coupled to the underlying ontology. Scaling the underlying ontology to incorporate larger concepts or relationships will not require overall changes in the indirect mapping components. As long as the main functionality of software components developed remains the same, changes in underlying ontology would only affect the relevant parts of the software.

Meanwhile, direct mapping and automatic generations of the object model are tightly coupled with the underlying information model (size). Changes in underlying ontology to incorporate larger number of concepts and relationship would have an immediate impact with the software components developed using direct mapping of ontology into object model.

7.3. Domain Changes Adaptation

A direct model is more suitable for a programmer writing domain-aware code; this is due to the fact that the object model generated in direct mapping retains the exact concepts and relationship as the original domain. In fact, software developers using the resulting object model are manipulating directly concepts and relationships in the original domain. Requests to apply the resulting direct software components into another domain will require major rewriting in the case of direct mapping approach.

An indirect model is instead more suitable for driving generic software that can be adopted in other domains. The indirect mapping might be more flexible in adapting to domain changes, as long as the newly adopted domains are compatible at the information model level, and a suitable conversion to internal representation of indirect mapping can easily be generated.

8. Recommendations for FI adopting ontologies

The results and observations made before in this article allow us to distill a set of recommendations for other projects in the Future Internet area that want to start with ontologies adoption. We do not have enough data to say if there is a clear advantage of using OWL ontologies in FI description and management. This choice is left to the Future Internet architects. But if, like in NOVI, the ontology adoption has already been chosen we can provide an answer to is what should be the mapping adopted in the FI software.

If we consider generically projects and software components our recommendations are:

1. Projects that start software and service development from scratch should chose direct mapping;
2. Projects whose software already exists (in part) will benefit more from indirect mapping;
3. Components providing services not exposed to external users can opt for indirect mapping;
4. Components that expose services to the outside should instead use direct mapping, in order to avoid continuos conversions.

These recommendations are in summarized in Table 1.

Project Scenarios	Indirect Mapping	Direct Mapping
Start from scratch		x
Existing services	x	
Internal communications		x
External communications	x	

Table 1

Recommendation Table

We can also provide recommendations considering the desired features of the software services, and the impact the mapping has on them. In the introduction we identified the main features of Future internet that information models need to support, namely common authorization policies, coordinated monitoring of resources, intelligent resource selection.

1. From the perspective of federating FI platforms direct mapping should be preferred. This is clear of we think of intelligent resource selection and discovery where there are heterogeneous resource providers. Direct mapping to the underlying ontology will simplify the information manipulation required to perform the discovery and selection task.
2. For monitoring which operates within a domain and only it is required to stitch information at the boundaries a direct mapping choice is less stringent.

9. Conclusion

The NOVI project has been one the first project to fully embrace semantic web as underlying mechanism to describe the concepts and elements required for the federation of Future Internet platforms. The motivation for ontologies is the additional reasoning feature which allows intelligent deduction of resource availability, policy and monitoring of infrastructure. In this article we have presented the ontologies that were developed in NOVI and summarised how they are used by the various services.

Of course the information model needs to be mapped to suite the needs of the federation services and the software components that will be developed. In this paper we have outlined different approach of performing this mapping. We described the concrete choices made in NOVI, namely direct and indirect mapping, and we gave the reader an insight in the problems encountered during the project that challenges the mapping choice.

We were able to generalise from our experience and we can conclude with a set of recommendations for upcoming projects in the area that are going to use ontologies in regard to the optimal mapping to be adopted.

Acknowledgments

The work presented here has been funded by the Dutch national project COMMIT.

Authors would also like to thank all the colleagues they worked with in the NOVI project (7th Framework Program, Grant No. 257867), and in particular Jeroen van der Ham and Chariklis Pittaras.

References

[1] Fed4FIRE. http://www.fed4fire.eu/.

[2] GENI - Exploring the Network of the Future. http://www.geni.net/.

[3] MOMENT Ű Monitoring and Measurement in the Next Generation Technologies. http://www.salzburgresearch.at/en/projekt/moment_en/.

[4] Guillaume Hillairet and Frédéric Bertrand and Jean Yves Lafaye. MDE for publishing Data on the Semantic Web. *Proceedings of the of the First Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE 2008) at MoDELS*, 2008.

[5] Vladan Devedzic. Understanding ontological engineering. *Communications of the ACM*, 45(4):136–144, 2002.

[6] DMTF. Common information model. http://dmtf.org/standards/cim.

[7] Sandor Laki and Blazej Pietrzak. NOVI deliverable 4.5: Integrated Prototype. http://www.fp7-novi.eu/deliverables/doc_download/71-d24, 2012.

[8] OpenRDF. Alibaba Version 2.0-rc7. http://www.openrdf.org/doc/alibaba/2.0-rc7/.

[9] Jeff Z Pan, Steffen Staab, Uwe ASSmann, Jürgen Ebert, and Yuting Zhao. *Ontology-Driven Software Development*. Springer Berlin Heidelberg, 2013.

[10] C. Pittaras, Papagianni. C., A. Leivadeas, P. Grosso, J. van der Ham, and S. Papavassiliou. Resource discovery and allocation for federated virtualized infrastructures. *Future Generation Computer Systems*, 2013.

[11] Colin Puleston, Bijan Parsia, James Cunningham, and Alan Rector. Integrating object-oriented and ontological representations: A case study in java and owl. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 130–145, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] Alan Rector, Sebastian Brandt, Nick Drummond, Matthew Horridge, Colin Pulestin, and Robert Stevens. Engineering use cases for modular development of ontologies in owl. *Appl. Ontol.*, 7(2):113–132, April 2012.

[13] John Strassner, José Neuman Souza, Sven Meer, Steven Davy, Keara Barrett, David Raymer, and Srini Samudrala. The design of a new policy model to support ontology-driven reasoning for autonomic networking. *J. Netw. Syst. Manage.*, 17(1-2):5–32, June 2009.

[14] J. van der Ham, Freek Dijkstra, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat. A distributed topology information system for optical networks based on the semantic web. *Optical Switching and Networking*, 5(2-3):85–93, 2008.

[15] Jeroen van der Ham. NOVI deliverable 2.4: Final Information and Data Model and Report on Prototypes. http://www.fp7-novi.eu/deliverables/doc_download/71-d24, 2011.

[16] Jeroen van der Ham. NOVI deliverable 2.1: Information models for virtualized architectures. http://www.fp7-novi.eu/deliverables/doc_download/25-d21, 2012.

[17] Jeroen van der Ham, Chrysa Papagianni, Jozsef Steger, Peter Matray, Yiannos Kryftis, Paola Grosso, and Leonidas Lymberopoulosà. Challenges of an information model for federating virtualized infrastructures. 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud, 2011.

[18] Adianto Wibisono, Ralph Koning, Paola Grosso, Adam Belloum, Marian Bubak, and Cees de Laat. Ointed: Online ontology instance editor enabling a new approach on ontology development. *Software Practice and Experience*, 43:1319–1335, 2013.