Evaluating Formalisms for Modular Ontologies in Distributed Information Systems

Yimin Wang[1], Jie Bao[2], Peter Haase[1], and Guilin Qi[1]

[1] Institute AIFB, University of Karlsruhe (TH), Karlsruhe, Germany
{ywa,pha,gqi}@aifb.uni-karlsruhe.de
[2] Artificial Intelligence Research Laboratory, Department of Computer Science
Iowa State University, Ames, IA 50011-1040, USA
baojie@cs.iastate.edu

Abstract. Modern semantic technology is one of the necessary supports for the infrastructure of next generation information systems. In particular, large international organizations, which usually have branches around the globe, need to manage web-based, complex, dynamically changing and geographically distributed information. Formalisms for modular ontologies offer the necessary mechanism that is needed to handle ontology-based distributed information systems in the aforementioned scenario. In this paper, we investigate state-of-the-art technologies in the area of modular ontologies and corresponding logical formalisms. We compare different formalisms for modular ontologies in their ability to support networked, dynamic and distributed ontologies, as well as the reasoning capability over these ontologies. The comparison results show the strength and limitation of existing formalisms against the needs of modular ontologies in the given setting, and possible future extensions to overcome those limitations.

Keywords: Distributed information systems, modular ontologies, semantic technology, requirements.

1 Introduction

Managing large-scale information systems in a distributed way is usually a challenging task in which each system may pertain only a subset of the information in question and the dynamically-changing information in local systems is difficult to detect or to control. A typical application scenario is that large international organizations, which may have branches around the globe and maintain multiple, distributed, large information systems for each of their branch. With the popularity of semantic technologies deployed in information system engineering, knowledge, often being represented as ontologies, is typically maintained by local branches of the organization in a collaborative way. While those ontologies are usually focused on the local information of particular local branches and are physically distributed around the world, they are also very likely to be linked together to offer the necessary global usage of information.

As a motivating example, we consider one of the case studies of the NeOn project[1] [8] – a fishery case study in the Food and Agriculture Organization (FAO) of the United

[1] http://www.neon-project.org

Nations (UN). This case study aims to improve the interoperability of FAO information systems in the fishery domain, integrating and using *networked ontologies* [14], by creating and maintaining distributed ontologies (and ontology mappings) in the fishery domain. In particular, FAO has large sets of fishery ontology data with the following features and requirements:

1. *Networking.* The ontology data sets in FAO are intensively interconnected by different subjects, languages, countries and other geopolitical aspects in a secure networked environment with clear boundary for information hiding and encapsulation.
2. *Dynamics.* In FAO, the ontologies are large, interconnected and changing over time. Therefore, an approach that can handle ontology data in a dynamic way with change monitoring and propagation is required.
3. *Distribution.* Because FAO has many branches around the world, the ontologies in FAO are distributed rather than centralized, which arises challenges in loose coupling and autonomous management.
4. *Reasoning.* FAO ontologies, which usually consist of both terminologies and assertional data in FAO fishery case studyies, need reasoning support with high efficiency and good scalability.

We argue that such a scenario is common to typical large-scale distributed information systems that are deployed using semantic applications, especially for knowledge management in big international organizations. Hence, there have been considerable recent efforts to provide solutions for such a scenario. For example, the recent W3C recommendation, OWL Web Ontology Language [23] can represent and connect ontologies on the Web in a machine readable format, which is one of the central concerns of the Semantic Web [5,30]. Borgida and colleagues proposes Distributed Description Logics (DDL) to correspond the federated information sources [6] and a DDL implementation DRAGO [28]. However, these current technologies often have difficulties in handling ontological data against the requirements mentioned above:

1. Traditional ontology formalisms, e.g. Frame System or Description Logics, are designed for centralized ontologies rather than decentralized ones. Furthermore, most ontology management systems do not support processing large instance data represented in the form of ontologies in the decentralized scenario.
2. In a scenario with interconnected ontology modules, ideally, when one ontology module is updated, the depending ontology modules should be updated as well to reflect the changes. However, few current technologies can support such dynamic automatic updating.
3. What is still missing is a *principled* approach to support distributed ontologies, where the individual ontology modules are physically distributed, loosely coupled and autonomously managed.
4. Some reasoners are able to support either local TBox reasoning (e.g. FaCT++ [35]) or distributed TBox reasoning (e.g. DRAGO [28] and Pellet [31]), while others are good at ABox reasoning (e.g. KAON2 [24]). However, handling both TBox and ABox in a distributed, efficient and scalable way for modular ontologies is a challenging task for reasoners.

To manage information generated and maintained in such distributed settings, we need knowledge representation formalisms and tools to meet the following challenge:

How to properly manage multiple networked, distributed, dynamic ontologies and provide corresponding reasoning support? In this paper, we investigate approaches to represent and exploit such networked ontologies, considering the recent advances in formalisms for modular ontologies and distributed reasoning techniques.

In the following sections, we firstly introduce some preliminaries of description logics and modular ontologies in Section 2. We analyze requirements of networked ontologies and comparison criteria for different formalisms on their language functionality and expressivity in Section 3. We compare several formalisms for the need of networked ontologies in the light of the given set of requirements in Section 4 and identify some critical unsolved problems in modeling and reasoning with networked ontologies and discuss possible solutions to those problems Section 6. We summarize the paper and outline future work in Section 7.

2 Preliminaries

Formalisms of modular ontologies studied in this paper mainly aim to handle subsets of OWL-DL with Description Logics (DLs) as the underlying logical formalism. Therefore, we first introduce basic preliminaries of DLs to allow a better understanding of the formalisms introduced in Section 4. Furthermore, we also briefly introduce modular ontologies and their roles in the distributed information systems.

2.1 Description Logics

Syntax. Given $\mathcal{R}$ as a finite set of transitive and inclusion axioms with normal role names N_R, a *$\mathcal{SHIQ}$-role* is either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$. $\mathsf{Trans}(R)$ and $R \sqsubseteq S$ represent the transitive and inclusion axioms, respectively, where R and S are roles. A simple role is a *$\mathcal{SHIQ}$-role* that neither its sub-roles nor itself is transitive. Let N_C be a set of *concept names* , the set of $\mathcal{SHIQ}$-concepts is the minimal set such that every concept $C \in N_C$ is a $\mathcal{SHIQ}$-concept and for C and D are $\mathcal{SHIQ}$-concepts, R is a role, S a simple role and n a positive integer, then $(\neg C)$, $(C \sqcap D)$, $(C \sqcup D)$, $(\exists R.C)$, $(\forall R.C)$, $(\leqslant nSC)$ and $(\geqslant nSC)$ are also $\mathcal{SHIQ}$-concepts.

Therefore we have a knowledge base $\mathcal{KB}$ that is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ where $\mathcal{R}$ is the RBox, TBox $\mathcal{T}$ is a finite set of axioms representing the concept inclusions with the

Table 1. Semantics of $\mathcal{SHIQ} - \mathcal{KB}$

Interpretation of Concepts
$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
$(\exists R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \vert R^\mathcal{I}(x, C) \neq \emptyset\}$, $(\forall R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \vert R^\mathcal{I}(x, \neg C) = \emptyset\}$
$(\leqslant nS.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \vert \mathbf{N} R^\mathcal{I}(x, C) \leqslant n\}$, $(\geqslant nS.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \vert \mathbf{N} R^\mathcal{I}(x, C) \geqslant n\}$
Interpretation of Axioms
$(C \sqsubseteq D)^\mathcal{I} : C^\mathcal{I} \subseteq D^\mathcal{I}$, $(R \sqsubseteq S)^\mathcal{I} : R^\mathcal{I} \subseteq S^\mathcal{I}$
$(\mathsf{Trans}(R))^\mathcal{I}$: $\{\forall x, y, z \in \Delta^\mathcal{I} \vert R^\mathcal{I}(x, y) \cap R^\mathcal{I}(y, z) \to R^\mathcal{I}(x, z)\}$
$\mathbf{N}R$ is the number restriction of a set R and $R^\mathcal{I}(x, C)$ is defined as: $\{y \vert \langle x, y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$.

form $C \sqsubseteq D$, ABox $\mathcal{A}$ is a finite set of axioms with the form $C(x)$, $R(x, y)$, and $x = y$ (or $x \neq y$) that consists (un)equality-relations.

Semantics. The semantics of $\mathcal{KB}$ is given by the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, ^{\mathcal{I}'})$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and the function $^{\mathcal{I}'}$ in the Table 1 [17]. The satisfiability checking of $\mathcal{KB}$ in expressive DLs is performed by reducing the subsumption, and the reasoning over TBox and role hierarchy can be reduced to reasoning over only role hierarchy [16]. The interpretation $\mathcal{I}$ is the model of $\mathcal{R}$ and $\mathcal{T}$ if for each $R \sqsubseteq S \in \mathcal{R}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ and for each $C \sqsubseteq D \in \mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

2.2 Modular Ontologies

Syntax. A modular ontology usually contains a set of component theories (modules) from same or different languages, and a set of semantic connections between those component. Formally, an abstract modular ontology $\Sigma = \langle \{L_i\}, \{M_{ij}\}_{i \neq j} \rangle$ contains a set of modules Li, each is a TBox of a subset of $\mathcal{SHIQ}$, and a set of semantic connections M_{ij} between L_i and L_j for some $i \neq j$.

Two broad types of modular ontology languages have been studied [3]. The linking or mapping approach requires signatures of modules to be disjoint, i.e. $\mathsf{Sig}(L_i) \cap \mathsf{Sig}(L_j) = \emptyset$, for $i \neq j$; M_{ij} serves as the *mapping* between L_i and L_j. On the other hand, the importing approach allows modules to share terms. Formally, for a module L_i, a subset of its symbols $\mathsf{Loc}(L_i) \subseteq \mathsf{Sig}(L_i)$ is called L_i's *local signature*; the set of terms in $\mathsf{Ext}(L_i) = \mathsf{Sig}(L_i) \backslash \mathsf{Loc}(L_i)$ is called L_i's *external signature*; a term $t \in \mathsf{Loc}(L_i) \cap \mathsf{Ext}(L_j)$ $(i \neq j)$ is said an imported term of j. Hence, semantic connection M_{ij} in the importing approach only allows name reuse in the form of $L_i \xrightarrow{t} L_j$.

Semantics. An interpretation $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{i \neq j} \rangle$ of abstract modular ontology $\Sigma = \langle \{L_i\}, \{M_{ij}\}_{i \neq j} \rangle$, where $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, (.)^{\mathcal{I}_i} \rangle$ is the *local interpretation* of module L_i; *domain relation* $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is the interpretation for the semantic connection from L_i to L_j. It should be noted that two domains $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$ are *not necessarily* the same or are disjoint. A domain relation r_{ij} represents the capability of the module j to map the objects of $\Delta^{\mathcal{I}_i}$ into $\Delta^{\mathcal{I}_j}$. Some formalisms may allow multiple domain relations under names $\{\mathrm{R}_1, ...\mathrm{R}_m\}$ such that $r_{ij} = \bigcup_n r_{ij}^{\mathrm{R}_n}$.

Different modular ontology languages provide different solutions to model component theories and semantic connections between them. In section 4, we will further introduce several representative families of modular ontologies.

3 Criteria for Evaluation

In this section, we discuss two categories of criteria for evaluating modular ontology formalisms: functionality criteria is driven by the requirements of applications of large-scale distributed information system, while the expressivity criteria is driven by the requirements on language modelling ability.

3.1 Functionality

We measure language functionalities supported by different formalisms in four dimensions as we introduced in Section 1.

Networking

– *Encapsulation.* In the networked ontology setting, each ontology module may represent knowledge in a subset of the domain in question. Those modules are usually autonomously created and maintained, while they may also be inter-connected to form a larger knowledge base. In other words, such ontology modules can *encapsulate* knowledge sub-domains, where the local domain is a subset of the global domain. Syntactically, instead of having a single logic theory L, we may have a set of local theories $\{L_i\}$ such that axioms will have clearly defined provenance from one of the local theory.
– *Reusability (Inheritance).* Ontologies are very likely to be reused. Thus, formalisms for modular ontologies should also support managing different modules and identifying module dependencies. Formally, if module L_1 reuses L_2 and if a subsumption $C \sqsubseteq D$ is entailed by L_2, such that for every model $\mathcal{I}_2$ of L_2, $C^{\mathcal{I}_2} \subseteq D^{\mathcal{I}_2}$, then we will have for every model $\mathcal{I}_1$ of L_1, $C^{\mathcal{I}_1} \subseteq D^{\mathcal{I}_1}$.
  In particular, ontology modules should be also transitively reusable, that is, if a module X uses module Y, and module Y uses module Z, then apparently, module X should use module Z.
– *Authorization.* Many applications call for controllable access of knowledge due to copyright, privacy or safety concerns. Formalisms for modular ontologies may also support authorization features to ensure authorized creation and usage of ontology modules. For example, to enable multi-user access to ontologies, the language supported by the formalism may secure the session by explicitly defined rights for accessing or editing of each module. Formally, for certain agent, an ontology module L_i with authorization may be divided into a hidden part L_i^H and visible part L_i^V, such that there is a reasoner for L_i such that if axiom $\alpha \in L_i^H$, then no any combination of L_i^V and previous answers from the same reasoner will entail α.

Dynamics

– *Networked Ontology Dynamics.* The dynamic role of networked ontologies reflects the importance of monitoring and propagating ontology changes and updating. Such a requirement is closely related to ontology evolution, while it is more focused on the identification and updating of module changes rather than managing ontology versions. Assume concept C is in ontology module L_1 and concept D is in module L_2, then we have can axioms to represent that C corresponds to D by certain intermodule correspondences. The dynamics of network ontology requires the changes on C to $C^{'}$ should be detectable by L_2 in order to preserve the global $(L_1 \sqcup L_2)$ consistency or satisfiability by changing D change to $D^{'}$ if necessary [32].

Distribution

– *Loose Coupling.* Modules in a modular ontology may only be loosely coupled, such that interconnections between different modules are well controlled, conflicts can be easily detected and eliminated, and communication cost in the reasoning process is minimized. On the other hand, two modules are not necessarily fully disjoint from each other. Syntactically, it can be measured by the "*connectedness*" notion [25], such that the size of axioms in mappings $\{M_{ij}\}$ or in component logics $\{L_i\}$ that contains terms from different components are minimized.
– *Self-Containment.* On the other hand, ontology modules may also be semantically loosely coupled such that a module could be self-contained in the sense that reasoning tasks may be locally preformed using only local knowledge when there is no required access for knowledge in other modules. It may be measured by the supporting to the "conservative extension"[22,11] property of ontology module, such that for any module L_i and L_j, for a query α in the language of L_i, $L_i \cup L_j \models \alpha$ iff $L_i \models \alpha$, i.e. the combination of logic modules will not change the internal knowledge structure of any module.

Reasoning

– *Complexity and Scalability.* Processing modular ontologies is typically more challenging than reasoning with a single ontology. Thus, a desirable formalism for modular ontologies should provide reasoning procedures that are efficient and scalable to large terminologies and instance sets. For example, the formalism should be able to scale when processing large number of modular ontology that are distributed.
– *Reasoning Support for Terminological and Assertional Knowledge.* A desirable feature for a modular ontology formalism is to supports both T-Box reasoning and ABox querying. Support for modular ABoxes is particularly important since our motivation applications involve knowledge that is represented in large and distributed instance sets.

3.2 Expressivity

Expressivity criteria of modular ontology formalisms include the following:

– *Module Correspondence.* As we mentioned in the functionality criteria, different modules may be partially coupled in their languages or interpretations. For example, an ontology module about academic department may borrow (inherit) some knowledge from another university ontology, which may be stated as:

$$\mathsf{DepartmentModule}\ isInheritedFrom\ \mathsf{UniversityModule}$$

– *Concept Subsumption.* Having subsumption relations between concepts in different modules is one of the most needed features in modular ontologies. For example, MasterStudent in a university ontology module may be a subclass of Student in the people ontology module.
– *Concept Interconnection.* A concept in a module may be connected to concepts in foreign modules by roles. For example, PhDStudent in the university module may be related to the City concept from a geographic ontology module by $lives$ role connection.

- *Role Subsumption.* It is used to allow the subsumption relationship between the local and the foreign role.
- *Role Transitivity.* A foreign transitive role may be used in a module, e.g., the module A reuses the property biggerThan which is defined in the module B.
- *Role Inversion.* It is required to specify inverse relations between local and foreign roles.
- *Individual Correspondence.* It is required to specify that some individuals in one module are related to individuals in other modules.

Given the above set of criteria on language functionality and expressivity, we will analyze candidate formalisms with details in the next section. Apparently, not all applications need all of the criteria above, which are mainly driven by the FAO fishery case study setting, nevertheless we still argue that this setting can be generalized and applied to many real world distributed information systems.

4 Candidates of Formalisms for Modular Ontologies

4.1 Modularization with OWL Import

The OWL ontology language provides limited support for modularizing ontologies: an ontology document – identified via its ontology URI – can be imported by another document using the *owl:imports* statement. The semantics of this import statement is that all definitions of the imported ontology (module) are logical part of the importing ontology (module) as if they were defined in the importing ontology directly. Thus, they are forced to share a classical DL semantics, i.e., a global model semantics. It should be noted that such an importing is directed: only the importing ontology is affected by the import statement; it is also transitive: if ontology A imports ontology B, and ontology B imports ontology C, then ontology A also imports ontology C. Cyclic imports are also allowed (e.g. A *owl:imports* B, B *owl:imports* A).

Terms of the importing and imported ontology module can be related to each other using legal primitives available in OWL. Typically these relation definitions are part of the importing ontology module.The *owl:imports* functionality provides no partial importing of modules, thus it is up to the user to decide the proper level of granularity of ontology modules.

4.2 Distributed Description Logics

Distributed Description Logics (DDL) [6] adopt a linking mechanism. In DDL, the *distributed knowledge base (D-KB)*, $\mathfrak{D} = \langle \{L_i\}_{i \in I}, \mathfrak{B} \rangle$ consists a set of local knowledge bases $\{L_i\}_{i \in j}$ and bridge rules $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{\{i \neq j\} \in I}$ that represent the correspondences between them. The semantic linkings between modules L_i and L_j are represented by cross-module *Bridge Rules* “INTO” and “ONTO” axioms in one of the following forms:

- INTO: $i : C \xrightarrow{\sqsubseteq} j : D$, with semantics: $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$
- ONTO: $i : C \xrightarrow{\sqsupseteq} j : D$,with semantics: $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$

where $\mathcal{I}_i$ and $\mathcal{I}_j$ are local interpretations of L_i and L_j, respectively, C, D are concepts, r_{ij} (called *domain relation*) is a relation that represents an interpretation of B_{ij}.

DDL bridge rules between concepts covers one of the most important scenarios in modular ontologies. They are intended to simulate concept inclusion with a special type of roles. However, a bridge rule cannot be read as concept subsumption, such as $i : A \sqsubseteq j : B$. Instead, it must be read as a classic DL axiom in the following way [6]:

- $i : A \xrightarrow{\sqsubseteq} j : B \Rightarrow (i : A) \sqsubseteq \forall R_{ij}.(j : B)$
- $i : A \xrightarrow{\sqsupseteq} j : B \Rightarrow (j : B) \sqsubseteq \exists R_{ij}^{-}.(i : A)$

where R_{ij} is a new role representing correspondences B_{ij} between L_i and L_j.

Such relations have semantic differences with respect to concept inclusion (interpreted in classic DLs as subset relations between concept interpretations, e.g. $A^{\mathcal{I}_i} \subseteq B^{\mathcal{I}_j}$) in several ways. For example, empty domain relation r_{ij} is allowed in the original DDL proposal [6], while GCIs between satisfiable concepts enforce restrictions on non-empty interpretations. Arbitrary domain relations may not preserve concept unsatisfiability among different modules which may result in some reasoning difficulties [3]. Furthermore, while subset relations (between concept interpretations) is transitive, DDL domain relations are not transitive, therefore bridge rules cannot be *transitively reused* by multiple modules. Those problems are recently recognized in several papers [2,3,34,27] and it is proposed that arbitrary domain relations should be avoided. For example, domain relations should be one-to-one [27,3] and non-empty [34].

The requirements of practical applications raised in the previous section are not fully satisfied by the expressivity of DDL. For example, inter-module role correspondences, which are important to present relations between concepts in different modules, are not supported in DDL: assume an concept PhDStudent is included in one ontology module and another concept Thesis is include in another ontology module, we cannot define PhDStudent $\sqsubseteq \exists$writes.Thesis in DDL, where writes is a *inter-module* role.

4.3 Integrity and Change of Modular Terminologies in DDL

Influenced by DDL semantics, Stuckenschmidt and Klein [32] adopt a view-based information integration approach to express relationships between ontology modules. In particular, in this approach ontology modules are connected by correspondences between conjunctive queries. This way of connecting modules provides a tradeoff between the simplicity of one-to-one mappings between concept names and the unrestricted use of logical languages to connect different modules.

Stuckenschmidt and Klein [32] defines an ontology module – abstracted from a particular ontology language – as a triple $M = (\mathcal{C}, \mathcal{R}, \mathcal{O})$, where $\mathcal{C}$ is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. A conjunctive query Q over an ontology module $M = (\mathcal{C}, \mathcal{R}, \mathcal{O})$ is defined as an expression of the form $q_1 \wedge ... \wedge q_m$, where q_i is a query term of the form $C(x)$, $R(x, y)$ or $x = y$, C and R are concept and role names, respectively, and x and y are either variable or object names.

In a modular terminology it is possible to use conjunctive queries to define concepts in one module in terms of a query over another module. For this purpose, the set of concept definitions $\mathcal{C}$ is divided into two disjoint sets of internally and externally defined concepts $\mathcal{C}_I$ and $\mathcal{C}_E$, respectively, with $\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_E, \mathcal{C}_I \cap \mathcal{C}_E = \emptyset$. An *internal concept* definition is specified using regular description logics based concept expressions with the form of $C \sqsubseteq D$ or $C \equiv D$, where C and D are atomic and complex concepts, respectively. An *external concept* definition is an axiom of the form $C \equiv M : Q$, where M is a module and Q is a conjunctive query over M. It is assumed that such queries can be later reduced to complex concept descriptions using the query-rollup techniques from [19] in order to be able to rely on standard reasoning techniques. A modular ontology is then simply defined as a set of modules that are connected by external concept definitions. The semantics of these modules is defined using the notion of a distributed interpretation introduced in Section 4.2.

Although the definition of a module, in its abstract form shown above, may allow arbitrary concept, relation and object definitions, only concept definitions is studied in [32]. This is due to the focus of the approach to improve terminological reasoning with modular ontologies by pre-compiling implied subsumption relations. In that sense it can be seen as a restricted form of DDLs that enables improved efficiency for special TBox reasoning tasks.

4.4 $\mathcal{E}$-Connection

While DDL allows only one type of domain relations, the $\mathcal{E}$-connection approach allows multiple "connections" between two modules, such as liveIn and bornIn between 2 : Fishkind and 1 : Region, where "2" and "1" stand for different modules, respectively. $\mathcal{E}$-connections between DLs [20,13] divide roles into disjoint sets of *local roles* (connecting concepts in one module) and *links* (connecting inter-module concepts). Formally, given ontology modules $\{L_i\}$, a (one-way binary) link $E \in \mathcal{E}_{ij}$, where $\mathcal{E}_{ij} (i \neq j)$ is the set of all links from the module i to the module j, can be used to construct a concept in module i, with the syntax and semantics specified as follows:

- $\exists E.(j : C) : \{x \in \Delta_i | \exists y \in \Delta_j, (x, y) \in E^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$
- $\forall E.(j : C) : \{x \in \Delta_i | \forall y \in \Delta_j, (x, y) \in E^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}\}$
- $\leq nE.(j : C) : \{x \in \Delta_i | \mathbf{N}(\{y \in \Delta_j | (x, y) \in E^{\mathcal{I}}, y \in C^{\mathcal{I}}\}) \leq n\}$
- $\geq nE.(j : C) : \{x \in \Delta_i | \mathbf{N}(\{y \in \Delta_j | (x, y) \in E^{\mathcal{I}}, y \in C^{\mathcal{I}}\}) \geq n\}$

where C is a concept in L_j, with interpretation $C^{\mathcal{I}} = C^{\mathcal{I}_j}$; $E^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is the interpretation of a $\mathcal{E}$-connection E; and $\mathbf{N}$ is the cardinality of set; $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{E^{\mathcal{I}}\}_{E \in \mathcal{E}_{ij}} \rangle$ is an interpretation of the $\mathcal{E}$-connected ontology, $\mathcal{I}_i$ is the local interpretation of L_i.

Existing $\mathcal{E}$-connection proposals [20,13] has required both local languages and local domains of ontology modules to be disjoint, which lead to several difficulties:

- The requirement for terminology disjointness and local domain disjointness in $\mathcal{E}$-connections enforce strong restrictions in some applications. For example, $\mathcal{E}$-connections is not able to refine role definitions in an existing module with a new module, e.g. i : hatchedIn is less general than j : bornIn, where j : bornIn is a role in an existing module and i : hatchedIn is a new role extended from j : bornIn.
- To enforce local domain disjointness, a concept cannot be declared as subclass of another concept in a foreign module thereby ruling out the possibility of asserting

inter-module subsumption and the general support for transitive usability; a property cannot be declared as sub-relation of a foreign property; neither foreign classes nor foreign properties can be instantiated; cross-module concept conjunction or disjunction are also illegal.
- $\mathcal{E}$-connected ontologies have difficulties to be used with OWL importing mechanism, since importing may actually "decouple" the combination and result in inconsistency [10].
- $\mathcal{E}$-connected ontologies do not allow a same term be used as both a link name and a local role name, nor role inclusions between links and roles, while such features are widely required in practice [10]. The"punning" approach [10], where a same name can have different interpretations, is rather as a syntactical sugar than a semantic solution to such problems.

4.5 Package-Based Description Logics

Package-based Description Logics (P-DL) [4], uses importing relations to connect local modules. In contrast to OWL, which forces the model of an imported ontology to be completely embedded in a global model, the P-DL importing relation is *partial* in that only commonly shared terms are interpreted in the overlapping part of local models. The semantics of P-DL is given as the follows: the *image domain relation* between local interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ (of package P_i and P_j) is $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. P-DL domain relation is:

- one-to-one: for any $x \in \Delta_i$, there is at most one $y \in \Delta_j$, such that $(x, y) \in r_{ij}$, and vice versa.
- compositionally consistent: $r_{ij} = r_{ik} \circ r_{jk}$, where $\circ$ denotes function composition. In other words, domain relations in P-DL is transitive.

P-DL provide contextualized semantics such that different packages have contextualized top concepts $\top_i$ (for all i) instead of a universal top $\top$; for any concept C, $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$. Hence, axiom in a P-DL package P_i will only be interpreted in its domain $\Delta^{\mathcal{I}_i}$, and may be influence only the overlapped domain $r_{ij}(\Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{I}_j}$ of another package. Therefore, knowledge in P-DL can be reused as well as keeping its contextuality.

P-DL also supports selective knowledge sharing by associating ontology terms and axioms with "scope limitation modifiers (SLM)". A SLM controls the visibility of the corresponding term or axiom to entities on the web, in particular, to other packages. The scope limitation modifier of a term or an axiom t_K in package K is a boolean function $f(p, t_K)$, where p is a URI of an entity, the entity identified by p can access t_K iff $f(p, t_K) = true$. For example, some representative SLMs can be defined as follows:

- $\forall p, public(p, t) := true$, means t is accessible everywhere.
- $\forall p, private(p, t) := (t \in p)$, means t is visible only to its home package.

P-DL semantics ensure that distributed reasoning with a modular ontology will yield the same conclusion as that obtained by a classical reasoning process applied to an integration of the respective ontology modules [3]. Reported result in [1] only supports reasoning in P-DL as extensions of $\mathcal{ALC}$ TBox. Reasoning algorithms for more expressive P-DL TBox and ABox reasoning still need to be investigated.

5 Evaluation

In this section, we first evaluate existing modular ontology formalisms, then we explain the results based on the evaluation criteria given in Section 3.

Table 2 evaluates different formalisms against the functionality requirements, and Table 3 compares the expressivity of the these formalisms. In this section, the integrity and change aspects related to modular ontologies investigated in [32] will be denoted as “DDL-IC”, since it follows the semantics of DDL.

– *Encapsulation.* According to our criteria, All formalisms listed above support knowledge encapsulation at different level. OWL- DL provides a basic *owl:import* primitive to import foreign ontologies without formal encapsulated modules, hence OWL-DL partially supports this functionality; DDL, DDL-IC, $\mathcal{E}$-connection and P-DL allow a large knowledge base to be represented by a set of ontology modules each capturing a subset of the domain of interest, thus provide the support for knowledge encapsulation.
– *Reusability*. OWL-DL ontologies has only limited reusability, because it is difficult for users to partially reuse ontologies designed by others. DDL, DDL-IC and P-DL establish good reusability via well-defined encapsulation and their semantics also satisfy with our criteria. The reusability of $\mathcal{E}$-connection is marked with a “*” symbol because the experiment in [26] shows for some knowledge bases, $\mathcal{E}$-connection is not able to generate reusable ontology modules. The lack of support for inter-module concept inclusion presents restriction in reusing $\mathcal{E}$-Connection modules. By combining the scope limitation and importing mechanism provided by P-DL, ontology modules may be reused through selected “interface” which is similar to that of code reusing in software engineering (the “+” symbol in the Table 2 means this additional feature).
– *Authorization*. Bao et.al. [4] show that it is possible to integrate authorization information with modular ontologies to guarantee the secure access, editing and reasoning with ontology modules. To the best of our knowledge, there is no other reported formalisms support this functionality with semantics described in the criteria.

Table 2. Comparison on language functionality. “T” means this formalism supports terminological (TBox) reasoning and “A” stands for the assertional (ABox) reasoning support. We refer the reader to the corresponding analysis for explanations of the “+” and “*” symbols.

	OWL-DL	DDL	DDL-IC	P-DL	$\mathcal{E}$-Connection
Encapsulation	Partial	Yes	Yes	Yes	Yes
Reusability	Fair	Good	Good	Good^{+}	Good*
Authorization	No	No	No	Partial	No
Ontology Dynamics	Yes	No	Yes	Unclear	Unclear
Loose Coupling	No	Yes	Yes	Yes	Yes
Self-Containment	Partial	Yes	Yes	Partial	Yes
Scalability	Low	Fair	Fair	Fair	Low
Reasoning Support	T and A	T and Partial A	T	T	T

– *Ontology Dynamics*. There have been rich study with respect to the dynamics of OWL-DL [15], and DDL- IC developed a mechanism to monitor the changes of modular ontologies [32], but it is not clear that other formalisms support this functionality.
– *Loose Coupling*. It is supported at different levels by different formalisms except for OWL-DL. Stuckenschmidt and colleagues explicitly argued its importance and deployment in DDL-IC [32].
– *Self-containment*. Due to the lack of localized semantics, OWL-DL does not fully support knowledge self-containment. In particular, reasoning in an OWL ontology requires the integration of all directly or indirectly imported ontologies of the given ontology. DDL-IC [32] introduces the self-containment functionality based on traditional DDL, while $\mathcal{E}$-connections requires strict separation of knowledge terminologies of ontology modules as well as their local interpretation domains [10]. P-DL can maintain the autonomy of individual modules; however, since P-DL adopts a partial semantic importing approach, reasoning in a P-DL ontology may also depend on its imported ontologies. According to our arguments in Section 3, DDL, DDL-IC and $\mathcal{E}$-Connection provide full support to this functionality by preserving the local knowledge structure while combining with other foreign modules.
– *Scalability*. The worst time complexity of the four formalisms studied in this paper are all exponential [18,1,10,28] for standard reasoning tasks, thereby we mainly discuss the scalability of these formalisms in the distributed environment. DDL, DDL-IC and P-DL support reasoning in a distributed setting in which ontology modules can be kept strictly separate, thus the integration of component ontology modules is avoided to obtain higher scalability in handling large ontologies. On the other hand, the current reasoning strategy for $\mathcal{E}$-connection, which is implemented in the reasoner Pellet [31], adopts the "coloring" but not physical separation of tableaux of ontology modules, hence requires implicit ontology integration to a single location, which may deteriorate its scalability.
– *Reasoning Support*. Reasoning support for OWL-DL has been successfully implemented in several highly optimized reasoners, such as FaCT++[35], Pellet[31] and KAON2; in particular KAON2 is optimized for reasoning with large ABoxes. DDL recently supported large ABox reasoning in a limited form [29]. Other formalisms have reported the support for TBoxes [12] only.

In the following, we explain the expressivity comparison of the formalisms:

– DDL-IC and P-DL define modules that may be related to other modules before the integration, while other languages do not define correspondences between modules.
– All formalisms but DDL support concept interconnections across modules. OWL-DL allows arbitrarily complex relations between concepts in different ontologies (i.e. semantic connections have the same expressivity as that of the local ontology language in each module). On the other hand, other formalism restrict the expressivity of concept connections to obtain both localized semantics and decidability. DDL, DDL-IC support concept subsumptions. $\mathcal{E}$- connection does not allow cross-module subsumption relationships, but allows two concepts being connected by

Table 3. Comparison of expressivity

	OWL-DL	DDL	DDL-IC	P-DL	$\mathcal{E}$-Connection
Module Correspondence.	No	No	Partial	Partial	No
Concept Subsumption.	Yes	Yes	Yes	Yes	No
Concept Interconnection.	Yes	Yes	Yes	Yes	Yes
Role Subsumption.	Yes	Yes	Yes	No	No
Role Transitivity.	Yes	No	Yes	No	Partial
Role Inversion.	Yes	No	Yes	No	No
Individual Correspondence.	Yes	Yes	Yes	No	No

links. P-DL supports both inter-module concept subsumptions and inter-module concept connections by roles.

– Role subsumption is provided by DDL [9], DDL-IC and OWL-DL. $\mathcal{E}$-connection does not allow a same name being shared by links and roles, and it does not allow role inclusions between modules. Role transitivity and inversion are not supported by DDL, DDL-IC. Reported P-DL formalism [2] does not allow role name importing hence does not support inter-module role subsumption, inversion and the reuse of transitive roles.
– The predicate *owl:sameIndividualAs* in OWL-DL supports simple individual correspondence by predicate . Among other formalisms, only DDL has investigated individual correspondence between ontology modules [29]. $\mathcal{E}$-connections does not allow cross- module individual correspondence since local domains of ontology modules are strictly disjoint. Such a feature is also missing in reported P-DL formalism (which only allows concept importing among ontology modules). DDL-IC allows a view with no variable being defined, which may be used to establish individual correspondence between ontology modules.

6 Discussion

The survey in the previous sections shows that existing formalisms may provide solutions with strength and weakness on different aspects to meet the requirement from our motivated applications, i.e. the FAO fishery case study in NeOn project.

First of all, a commonly accepted definition of "What is a good ontology module?" is still missing. It has been argued that different application scenarios may require different set of modularity requirements [21]. Secondly, an efficient and scalable reasoning approach for large ABox data is currently not well provided by existing formalisms. Thirdly, most of the existing approaches can not support trust and authorization requirements. Finally, managing the dynamics of ontologies is only supported by the approach proposed by Stuckenschmidt and colleagues [33], which is still missing in other modular ontology formalisms.

Existing formalisms are also limited in expressivity. Most formalisms provide means to deal with concepts in terminology knowledge. However, mechanisms to handling other ontological entities, such as roles and individuals correspondences, is not supported by $\mathcal{E}$-connection or P-DL in their current forms. On the other hand, interconnections and relationships between modules, which is needed by many applications using modular ontologies, is currently not well-defined and lacks implementations.

Existing formalisms require further extensions, such as conservative extension [7], in order to be served as successful formalisms for modular ontologies as discussed in Section 3. Practical reasoning support for expressive formalisms for modular ontologies are also to be investigated.

7 Conclusions and Future Work

In this paper, we studied different formalisms for modular ontologies against the requirements within the context where deploys modern semantic technologies as a novel approach for large-scale distributed information system engineering. We presented a set of formal criteria based on the requirements of a typical networked ontology applications. We then compared several formalisms for modular ontologies against these requirements.

The comparison result suggests that no existing approach can satisfactorily meet all the requirements of our networked ontology applications. Several possible extension of existing formalisms were identified and discussed.

Work in progress includes the development of a networked ontology based formalism that meets the requirements raised in the Section 3, and the efficient and scalable reasoning support for such a formalism that is able to handle both large distributed ontology terminologies and instance data sets that are contained in the large-scale distributed information systems.

Acknowledgments

Research reported in this paper is partially support by the EU in the IST project NeOn (EU IST-2006-027595, `http://www.neon-project.org/`). We appreciate the fruitful discussion and comments from our colleagues.

References

1. Bao, J., Caragea, D., Honavar, V.: A distributed tableau algorithm for package-based description logics. In: the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006 (2006)
2. Bao, J., Caragea, D., Honavar, V.: Modular ontologies - a formal investigation of semantics and expressivity. In: Mizoguchi, R., Shi, Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 616–631. Springer, Heidelberg (2006)
3. Bao, J., Caragea, D., Honavar, V.: On the semantics of linking and importing in modular ontologies. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 72–86. Springer, Heidelberg (2006)
4. Bao, J., Caragea, D., Honavar, V.: Towards collaborative environments for ontology construction and sharing. In: International Symposium on Collaborative Technologies and Systems (CTS 2006), pp. 99–108. IEEE Press, Orlando (2006)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284, 34–43 (2001)

6. Borgida, A., Serafini, L.: Distributed description logics: Directed domain correspondences in federated information sources. In: OTM Federated Conference CoopIS/DOA/ODBASE. pp. 36–53 (2002)
7. Cuenca Grau, B., Kazakov, Y., Horrocks, I., Sattler, U.: A logical framework for modular integration of ontologies. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007) (2007)
8. Dzbor, M., Motta, E., Studer, R., Sure, Y., Haase, P., Gmez-Prez, A., Benjamins, R., Waterfeld, W.: Neon - lifecycle support for networked ontologies. In: Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT-2005), pp. 451–452 London, UK, IEE (2005)
9. Ghidini, C., Serafini, L.: Mapping properties of heterogeneous ontologies. In: 1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC (2006)
10. Grau, B.C.: Combination and Integration of Ontologies on the Semantic Web. PhD thesis, Dpto. de Informatica, Universitat de Valencia, Spain (2005)
11. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modular integration of ontologies. In: Proceedings of IJCAI'07 (2007)
12. Grau, B.C., Parsia, B., Sirin, E.: Tableau algorithms for e-connections of description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), TR 2004-72 (2004)
13. Grau, B.C., Parsia, B., Sirin, E.: Working with multiple ontologies on the semantic web. In: International Semantic Web Conference. pp. 620–634 (2004)
14. Haase, P., Rudolph, S., Wang, Y., Brockmans, S., Palma, R., Euzenat, J., d'Aquin, M.: D1.1.1 networked ontology model. NeOn Deliverable 1.1.1, Universität Karlsruhe, UPM, INRIA-ALPES, Open University (2007)
15. Haase, P., Sure, Y.: State-of-the-art on ontology evolution. Technical report, SEKT informal deliverable 3.1.1.b, Institute AIFB, University of Karlsruhe (2004)
16. Horrocks, I., Patel-Schneider, P.F.: Optimizing description logic subsumption. Journal of Logic and Computation 9, 267–293 (1999)
17. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web. Semantics 1, 7–26 (2003)
18. Horrocks, I., Sattler, U.: A Tableaux Decision Procedure for SHOIQ. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI'05. pp. 448–453 (2005)
19. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 399–404, AAAI Press / The MIT Press (2000)
20. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: E-connections of description logics. In: Description Logics Workshop, CEUR-WS, Vol 81 (2003)
21. Loebe, F.: Requirements for logical modules. In: 1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC (2006)
22. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence IJCAI-07, AAAI Press, Stanford (2007)
23. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C) (2003) Internet: http://www.w3.org/TR/owl-features/
24. Motik, B., Sattler, U.: A comparison of reasoning techniques for querying large description logic aboxes. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, Springer, Heidelberg (2006)

25. Schlicht, A., Stuckenschmidt, H.: Towards structural criteria for ontology modularizationc. In: 1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC (2006)
26. Seidenberg, J., Rector, A.: Web ontology segmentation: Analysis, classification and use. In: Proceedings of the World Wide Web Conference (WWW), Edinburgh (2006)
27. Serafini, L., Stuckenschmidt, H., Wache, H.: A formal investigation of mapping languages for terminological knowledge. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI'05, Edinburgh, UK (2005)
28. Serafini, L., Tamilin, A.: Drago: Distributed reasoning architecture for the semantic web. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 361–376. Springer, Heidelberg (2005)
29. Serafini, L., Tamilin, A.: Instance retrieval over a set of heterogeneous ontologies. In: the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006 (2006)
30. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems 21, 96–101 (2006)
31. Sirin, E., Parsia, B.: Pellet: An OWL DL Reasoner. In: Description Logics Workshop (2004)
32. Stuckenschmidt, H., Klein, M.C.A.: Integrity and change in modular ontologies. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence - IJCAI'03. pp.900–908 (2003)
33. Stuckenschmidt, H., Klein, M.C.A.: Structure-based partitioning of large concept hierarchies. In: International Semantic Web Conference. pp.289–303 (2004)
34. Stuckenschmidt, H., Serafini, L., Wache, H.: Reasoning about ontology mappings. Technical report, Department for Mathematics and Computer Science, University of Mannheim; TR-2005-011 (2005)
35. Tsarkov, D., Horrocks, I.: Efficient reasoning with range and domain constraints. In: Description Logics. FaCT++ (2004)