Article

JIS

Journal of Information Science
2015, Vol. 41(3) 296–314

DOI: 10.1177/0165551514560952
jis.sagepub.com
SAGE

Object-oriented and ontology-alignment patterns-based expressive Mediation Bridge Ontology (MBO)

Wajahat Ali Khan
Kyung Hee University, South Korea

Muhammad Bilal Amin
Kyung Hee University, South Korea

Asad Masood Khattak
Zayed University, UAE

Maqbool Hussain
Kyung Hee University, South Korea

Muhammad Afzal
Kyung Hee University, South Korea

Sungyoung Lee
Kyung Hee University, South Korea

Eun Soo Kim
Kwangwoon University, South Korea

Abstract
The Semantic Web is dependent on extensive knowledge management by interlinking resources on the web using matching techniques. This role is played by the progressing domain of ontology matching, by introducing ontology-matching tools. The focus of these matching tools is limited to matching techniques and automation, rather than expressive formal representation of alignments. We propose Mediation Bridge Ontology (MBO), an expressive alignment representation ontology used to store correspondences between matching ontologies matched by our ontology-matching tool, System for Parallel Heterogeneity Resolution (SPHeRe). The MBO utilizes object-oriented design patterns and the proposed ontology-alignment design patterns to provide extendibility and reusability factors to SPHeRe system. We compared our proposed system with existing systems using Coupling Factor, Number of Polymorphic methods and Rate of Change metrics to support extendibility and reusability. These factors contribute to the overall objective of interoperability for knowledge management in the Semantic Web.

Keywords
Design pattern; matching; interoperability; ontology; semantics

Corresponding author:
Sungyoung Lee, Department of Computer Engineering, Kyung Hee University, Seocheon-dong, Giheung-gu, Yongin-si, Gyeonggi-do446-701, Republic of Korea.
Email: sylee@oslab.khu.ac.kr

1. Introduction

The continuous evolution of heterogeneous data repositories is a major hindrance for Semantic Web infrastructure to facilitate mashup-like information sharing. Ontology matching has made measurable progress to resolve semantic heterogeneities among these heterogeneous data repositories for ontology merging, query answering or data translation [1]. The Ontology Alignment Evaluation Initiative[1] (OAEI), a benchmarking initiative, carries out annual campaigns for the evaluation of the ontology-matching tools [2]. OAEI has evaluated several ontology-matching systems; some of these remained in spotlight for many years. Shvaiko and Euzenat [1] present a survey of some of the recent matching systems based on their operations and matching approaches. A common behaviour among these matching tools is their duration of use that lasts for few years, with them being replaced after this time. The main reasons for their replacement is the difficulty in extending and reusing these systems. The structure of matching systems should be extendible enough to accommodate new algorithms based on novel matching techniques, replace previous algorithms if they are non-effective and utilize a combination of existing techniques to build new techniques. Therefore, incorporating object-oriented design patterns [3] with ontology design patterns in ontology-matching tools defines the longer adaption of such systems.

Ontology Design Patterns (ODP) support pattern-based ontology design [4] and are used to capture common modelling situations, help facilitate ontology development and avoid common mistakes [5]. ODP have evolved from Content Ontology Design Patterns (reusable solutions to recurrent content modelling problems) [6] to Ontology Alignment Design Patterns (used to refine correspondences, by alignment designer or pattern detection algorithm) [7]. Ontology matching algorithms detect simple correspondences by following an alignment format that lacks the expressiveness needed to formalize correspondence [8]. Therefore, an approach is necessary to design and develop an extendible and reusable system that provide expressive capability to formalize correspondences. The proposed Mediation Bridge Ontology (MBO) based approach incorporates object-oriented design patterns combined with ontology-alignment design patterns in our ontology-matching system, System for Parallel Heterogeneity Resolution (SPHeRe) [9].

The proposed MBO ontology is an ontology-alignment representation scheme that enables expressiveness to formalize correspondence by utilizing object-oriented and ontology-alignment design patterns. Existing ontology-matching schemes only focus on matching the ontologies and storing their alignments in a format that only describes source and target concepts. It is necessary to find the alignments using design patterns for providing solutions to the common problems. Also, expressiveness in the storage of correspondence is necessary for multiple reasons. First, expert verifications become easier as the correspondence speaks for itself. The correspondence includes not only the source and target concepts, but also the attributes involved in correspondence, the procedure of the alignments and the confidence value of the alignment. Second, feedback about the matching process and alignment can be easily obtained, which helps in the overall improvement of the system and satisfaction of the users. We developed the SPHeRe ontology-matching system that incorporates bridge algorithms that are stored in expressive alignment representation format in the MBO. Strategy[2] and Mediator[3] design patterns are used from object-oriented design patterns, combined with ontology-alignment patterns called Pattern Relationship Models (PRM). The PRMs are the ontology-alignment patterns that define the expressive formal representation of the correspondences to be stored in the MBO. The proposed system supports collaborative ontology concepts by adding metadata information in alignments stored in the MBO. This helps in achieving extendibility and reusability metrics of the overall SPHeRe system.

Benchmarking and systematic evaluation are still progressing in the area of semantic technologies [2]. To evaluate the proposed system's extendibility and reusability capabilities, we used a Quality Model for Object Oriented Design approach [10]. The factors that contribute to the lack of longer adaptation of the existing ontology-matching systems are coupling, polymorphism and the rate of change. Therefore, we used Coupling Factor (COF), Number of Polymorphic methods (NOP) and Rate of Change (RoC) to evaluate the system against the existing system to accomplish extendibility and reusability. This work contributes to the overall objective of interoperability among heterogeneous ontologies.

The rest of the paper is structured as follows: Section 2 addresses the methodologies used by existing ontology-matching systems and compares them with the proposed system. The proposed MBO design and development are described in Section 3. Section 4 explains the integrated approach of object-oriented design patterns and ontology-alignment design patterns based on MBO. Section 5 shows the working model of the system and describes the working mechanism of the different bridge algorithms to populate MBO. Section 6 evaluates the proposed system by calculating and comparing values of evaluation metrics with existing systems. Section 7 concludes the paper and provides information about future work.

2. Related work

Design patterns provide solutions to common occurring problems, and ontology domains utilize ontology design patterns to facilitate the ontology development process. One of the potential areas of semantic technologies that is focusing on

incorporating design patterns as the solution to semantic heterogeneity problems is ontology matching, which finds similarities between concepts. Peigang Xu et al. [11] proposed a differentor-based similarity matrix creation technique used to integrate different similarity measures. Weights are assigned to various entities of the matching ontologies that are used for aggregation tasks after finding the similarity measures. Another approach proposed Tree Structure Based Ontology Integration [12] methodology, used to integrate ontologies with Document Type Definition-based tree structure development for ontology mappings. This is further utilized by ontology applications for data sharing purposes. These approaches led to the development of ontology-matching tools/systems. OAEI provides a platform to introduce state-of-the art ontology-matching tools, but their adaption for limited years, difficulty in extendibility and reusability and expressive mapping representations define the future directions for ontology-matching tools. Some of these tools and approaches for ontology-alignment patterns are discussed in this section.

We selected some ontology-matching tools for discussion in this section based on their participation and adaption in OAEI, and also some of the existing state-of-the-art systems. Falcon is one of the ontology-matching systems that has shown the best results in the first few years of OAEI campaign [13]. It provides fundamental technologies for finding, aligning and learning ontologies [13] by using a divide-and-conquer approach to target large ontologies generating 1:*n* alignments as output [1]. Although this system is still effective in generating alignments between ontologies owing to its matching techniques and also the user interface, lack of extendibility and reusability is its major disadvantage. It is extremely difficult to add new matching techniques and algorithms in the system. Agreement Maker is another ontology-matching tool that resolves the extendibility issue by displaying the ontologies, supporting several mapping layers visually and presenting automatically generated mappings for producing the alignments [14, 15]. This system is not scalable for large-scale ontology matching, but provides a flexible and extensible framework with a comprehensive user interface. The scalability issue is resolved in its new framework AgreementMakerLight, which preserves the original Agreement Maker framework with the main focus on computational efficiency and handling very large ontologies [16]. AgreementMakerLight competes with the recent OAEI performers, GOMMA and LogMap, in a large bio-med track, but lacks an approach for expressive mapping representation.

GOMMA [17] provides infrastructure for managing the matching and evolution of ontologies and its impact on mappings. On the other hand, LogMap is an ontology-matching tool that address the scalability issue for large ontology-matching and produces almost clean sets of output mappings [18]. GOMMA and LogMap demonstrates better accuracy as compared with other systems and were equally matched by another matching tool, YAM++ . The YAM++ matching tool supports self-configuration, extensibility and extensibility in combining individual matchers [19]. It discovers mappings using information-retrieval techniques and also deals with multilingual ontology-matching problems [20]. Most of the ontology-matching systems focus on automation and accuracy of results and not on expressive alignment representation using ontology-alignment patterns. Šváb-Zamazal et al. [21] presented a generic framework for ontology pattern detection, generation of instructions and ontology transformation from source ontology to target ontology. Scharffe et al. [7] took a step forward by introducing ontology-alignment design pattern representation methods and then created a pattern library to be extended with new patterns. The work also explains the transformation of ontologies using ontology-alignment patterns. To summarize, existing ontology-matching tools and ontology-alignment pattern-based approaches are unable to reflect a comprehensive system that utilizes an object-oriented design pattern combined with ontology-alignment design patterns and storing the correspondences between matched ontologies into a mapping storage and representation repository. Our proposed MBO-based ontology-matching methodology addresses the existing systems issues in our SPHeRe ontology-matching system as an extendible, reusable and expressive mapping representation approach.

3. Mediation Bridge Ontology

Ontology mediation techniques provide the platform for interoperability between heterogeneous ontological descriptions [7]. Mediation is based on the alignments generated between heterogeneous sources, and representation of these alignments plays a vital role in effective interoperability. Little focus is provided on the alignment representation area by the Semantic Web community [22]. An effort towards representing correspondences as a centralized repository was introduced as bridge ontology [23, 24], but it lacked effectiveness, agility and realization. Although it provided the base for alignment representation, it was never the focus, mainly because accuracy of alignments is given higher priority. *Effective alignment representation results in: (a) efficient ontology translation; (b) format transformationsl (c) systems mediation and (d) easy expert verification and modification*. Therefore, the proposed MBO targets effective alignment representations in its design and development process. The design aspect utilizes object-oriented as well as ontology-alignment design patterns for effective mapping representation in the form of low coupling, high polymorphism and low rate of change. The MBO benefits not only the ontology-alignment storage, but also its use in the transformation process between different heterogeneous formats. The scope of MBO is categorized into three aspects: Generalized Mappings,

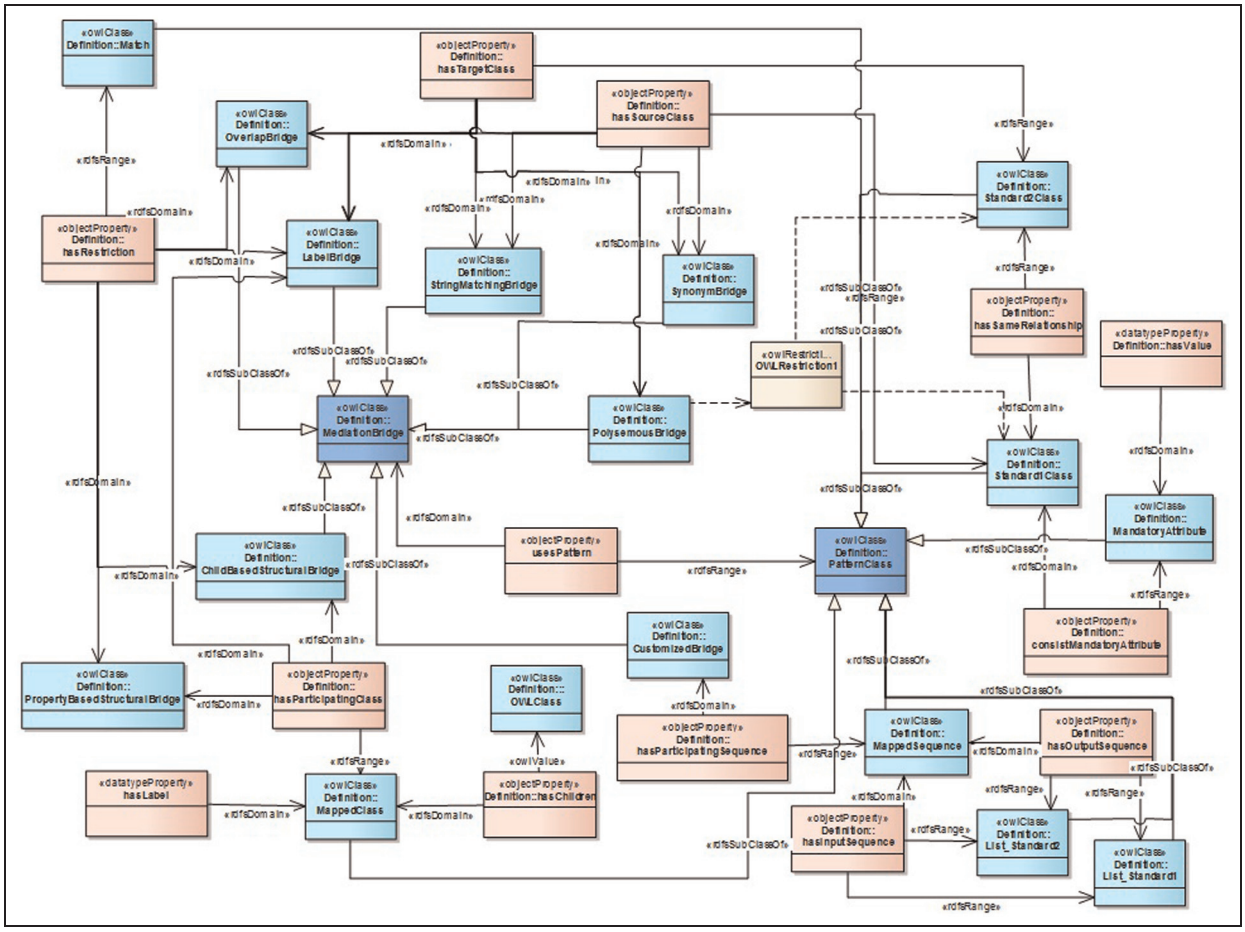


Figure 1. Mediation Bridge Ontology.

Customized Mappings and Transformation Logic. Generalized Mappings are the alignments that are generated by matching two ontologies using PRMs. The proposed MBO provides the alignment representation scheme using an ontology design patterns approach keeping in mind the goals expressed for achieving true interoperability. Customized Mappings are alignments that are based on the conformance issues handling of organizations. Organization conformance issues are handled through these mappings by detecting the stale mappings initially in the Generalized Mappings and then replacing them with the new modified mappings. The generalized as well as the customized mappings are converted into transformation logic that is used for conversion among different standard formats. The formal description of these concepts is provided in Section 3.2.

MBO is categorized into two main classes: *MediationBridge* and *PatternClass*. *MediationBridge* is divided into syntactic and structural bridge subclasses: *String Matching Bridge*, *Label Bridge*, *Synonym Bridge*, *Polysemous Bridge*, *Overlap Bridge*, *Customized Bridge*, *Children Based Structural Bridge and Property Based Structural Bridge*. These bridge classes are used to represent the alignments generated from particular algorithms in their specified format. These are dependent on *PatternClass* for structuring the output of the alignment process. *PatternClass* includes *MappedSequence*, *Standard1Class*, *Standard2Class*, *Match*, *MappedClass*, *ListStandard1* and *ListStandard2* subclasses. These are used to provide the structure for representation of the alignment in the MBO. The overall hierarchy of MBO is shown in Figure 1.

These concepts are related to each other using object properties; the triples are shown in Table 1. *MediationBridge* class is related through the *usesPattern* object property to *PatternClass*. Every subclass of *MediationBridge* uses some pattern classes from the *PatternClass* subclasses to define its alignment representation. *OverlapBridge* class is related through *hasSourceClass* and *hasTargetClass* object properties to *Standard1Class* and *Standard2Class*, respectively. *Standard1Class* uses *hasSameRelationship* and *consistsMandatoryAttributes* object properties to connect with *Standard2Class* and *MandatoryAttribute*, respectively. Based on the previous triples, *OverlapBridge* is related to the *Match* class using *hasRestriction* object property. This makes the complete alignment representation for *OverlapPRM*

Table 1. Mediation Bridge Ontology triples

Domain	Property	Range
Mediation Bridge	usesPattern	PatternClass
Standard1Class	exactMatch	Standard2Class
Customized Bridge	hasParticipatingSequence	MappedSequence
Label Bridge	hasSourceClass	Standard1Class
Overlap Bridge		
Polysemous Bridge		
String Matching Bridge		
Synonym Bridge		
Label Bridge	hasTargetClass	Standard2Class
Overlap Bridge		
Polysemous Bridge		
String Matching Bridge		
Synonym Bridge		
Mapped Class	hasChildren	owl:Class
Mapped Sequence	hasInputSequence	ListStandard1
		ListStandard2
Mapped Sequence	hasOutputSequence	ListStandard1
		ListStandard2
Label Bridge	hasRestriction	Match
Overlap Bridge		
PBSB		
CBSB		
Standard1Class	hasSameRelationship	Standard2Class
PBSB	hasParticipatingClass	MappedClass
CBSB		
Standard1Class	consistsMandatoryAttribute	MandatoryAttribute

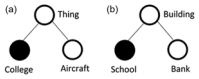
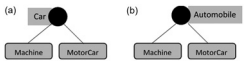
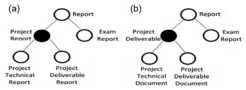
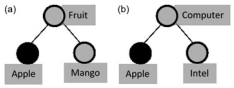
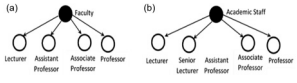
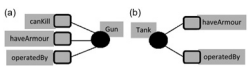
described in the later section. In the same way, other *MediationBridge* classes define their pattern to represent alignment in the MBO.

3.1. MBO bridges definition, examples and scenarios

The MBO provides the platform that represents alignments found by different bridge algorithms. These bridge algorithms are defined and explained with real-world examples, and scenarios using medical standard ontologies. We use medical standard ontologies as scenarios for the proposed system. The alignments generated and represented in the MBO can be used for ontology translation, standard format transformation and expert verification based on metadata availability about every alignment. One of the bridge algorithms is String Matching Bridge, which is used for concepts matching to identify similar concepts in the matched ontologies. These are based on string-based matching techniques by considering the sequence of letters of matching concepts. These sequences of letters considered for matching are based on the intuition that, the more similar the strings are, the more likely it is that the concepts will be similar [25]. Edit distance is one of the techniques used for string-based matching techniques [26]. Table 2 shows the examples and medical ontology scenarios of SPHeRe's bridge algorithms.

- Synonym Bridge (Table 2, row 2) represents identical or closely aligned concepts between different ontologies. College and School are two synonym concepts where College $\in O_i$ (O is ontology) and School $\in O_j$. Drug and Medicine are synonym concepts from SNOMED CT and Mesh standard ontologies as shown in Table 2.
- Label Bridge (Table 2, row 3) represents similar concepts based on common information represented as their labels. Car and Automobile are two similar concepts where Car $\in O_i$ and Automobile $\in O_j$. Car and Automobile both have Machine and Motorcar as their labels. Therefore, their similarity is based on the label match. Cartilage Cell concept of FMA ontology is similar to the Chondrocyte concept of NCI ontology based on a common label as Cartilage Cell.
- Overlap Bridge (Table 2, row 4) represents concepts that contain overlapping information that is necessary for data format transformation with information exchange between heterogeneous systems. Project Report and Project Deliverable are two overlapping concepts, containing most of the information common between them. Taking the example of HL7 and openEHR ontologies, OBSERVATION and EVALUATION are overlapping concepts.

Table 2. Mediation Bridge Ontology concepts, examples, scenarios

Bridge	Example	Medical ontologies scenario
Synonym Bridge		[SNOMED CT and Mesh ontology]: Concept DRUG of SNOMED CT ontology which is synonym of concept MEDICINE of Mesh ontology.
Label Bridge		[FMA and NCI ontology]: Concept CARTILAGE CELL of FMA ontology which is similar to concept CHONDROCYTE of NCI ontology based on common label CARTILAGE CELL.
Overlap Bridge		[HL7 and openEHR ontology]: OBSERVATION concept exists in both standard ontologies, and EVALUATION is the subconcept of OBSERVATION concept in openEHR ontology. Therefore, EVALUATION concept of openEHR ontology can also be transformed to OBSERVATION concept of HL7 ontology with information exchange.
Polysemous Bridge		[SNOMED CT and HL7 ontology]: EVENT concept in SNOMED CT ontology includes concepts that represent occurrences of different events while in HL7 ontology it is any act that has taken place. EVENT concept of SNOMED CT ontology and HL7 ontology are polysemous in nature.
Child Based Structural Bridge		[HL7 and openEHR ontology]: ENTITY concept of HL7 ontology is equivalent to PARTY concept in openEHR ontology based on their children being matched. ENTITY concept has ORGANIZATION, PERSON and DEVICE subconcepts that are mapped with ORGANIZATION, PERSON and AGENT subconcepts of the PARTY concept.
Property Based Structural Bridge		[HL7 and VMR ontology]: OBSERVATION concept belongs to HL7 ontology while OBSERVATION RESULT concept is part of VMR ontology. Both the concepts are similar based on property match. CODE, CODE SYSTEM, and DISPLAY NAME are the common properties between the concepts that lead to the conclusion of property-based match.

- Polysemous Bridge (Table 2, row 5) is used to cover the same concepts having different meaning cases during ontology matching. The concept Apple can represent a Fruit and it can also characterize Computer. Therefore, the Apple concept as a Fruit and as a Computer is an illustration of Polysemous bridge. Event concept in SNOMED CT and HL7 has different meaning with same concept name described in Table 2.
- CBSB (Table 2, row 6) represents concepts and relations/properties that are similar by comparing similarities between their children. Faculty and Academic Staff are two equivalent concepts based on children matching. In HL7 and openEHR medical standard ontologies, Entity (HL7) and Party (openEHR) are similar based on children matching.
- PBSB (Table 2, row 7) represents the concepts that are similar to each other based on their properties. Gun and Tank are two concepts of two separate ontologies similar to each other based on their common properties 'haveArmour' and 'operatedBy'. Similarly, Observation and ObservationResult concepts of HL7 and VMR ontologies are similar to each other based on their properties.

3.2. Formal modelling and representation of MBO

MBO formal modelling using Backus–Naur Form[4] is described in this section. MBO constructs are defined by the generalized and customized mappings which are then represented in logic format for transformation among different standards. The generalized mappings are the focus of this paper and include the alignment information with the ontology-alignment design pattern used for the creation of generalized mappings logic to be used for transformation. The formal definitions of all these concepts as well as the transformation logic based on generalized mappings are presented as follows:

$$< MBO > ::= \text{“}Generalized\ Mappings : \text{”} < GM >$$
$$\text{“}Customized\ Mappings : \text{”} < CM >$$
$$\text{“}Transformation\ Logic : \text{”} < Logic >$$

$$< GM > ::= \text{“}Alignment\ Info : \text{”} < Alignlnfo > \text{“Pattem Relationship Model : ”} < \text{PRM} > \text{“}Logic\ GM : \text{”} < LogicGM >$$

$$< Alignlnfo > ::= \text{“}Source\ Entity : \text{”} < SE > \text{“Target Entity : ”} < \text{TE} > \text{“}Measure\ Threshold\ Value : \text{”} < MTV > \text{“Relationship : ”} < \text{R} >$$

$$< SE > ::= \{x | O_1 \cap x \in < S_\Delta, x_i > \}$$

$$< S_\Delta, x_i > ::= \{(x_i \in S_\Delta) \wedge (S_\Delta \in O_1)\}$$

$$< TE > ::= \{x | O_2 \cap x \in < T_\Delta, x_i > \}$$

$$< T_\Delta, x_i > ::= \{(x_i \in T_\Delta) \wedge (T_\Delta \in O_2)\}$$

$$< MTV > ::= \{(\exists SE_\Delta \leftarrow O_1) \leftrightarrow (\exists TE_\Delta \leftarrow O_2) \cap (x | x\ is\ a\ threshold\ value)\}$$

$$< R > ::= \{(\exists SE_\Delta \leftarrow O_1) \leftrightarrow (\exists TE_\Delta \leftarrow O_2) \cap (x | x\ is\ relationship\ between\ SE\ and\ TE)\}$$

$$< PRM > ::= StringPRM | ChildPRM | LabelPRM | PropertyPRM | OverlapPRM | CustomizedPRM | SynonymPRM | PolysemousPRM$$

$$< LogicGM > ::= < Logic1 > < Logic2 > \cdots < LogicN >$$

$$< Logic1 > ::= TE \leftarrow SE$$

$$< Logic2 > ::= \{TE \cap \{\exists TE.\ attribute \wedge (TE.\ attribute \geq 1)\}\} \leftarrow SE$$

$$< Logic3 > ::= TE \leftarrow \{SE \cap \{\exists SE.\ attribute \wedge (SE.\ attribute \geq 1)\}\}$$

$$< Logic4 > ::= \{TE \cap \{\exists TEChild \subseteq TE \wedge (TEChild \geq 1)\}\} \leftarrow SE$$

$$< Logic5 > ::= TE \leftarrow \{SE \cap \{\exists SEChild \subseteq SE \wedge (SEChild \geq 1)\}\}$$

$$< Logic6 > ::= TE \leftarrow \{SE \cap \{(\exists SEChild \subseteq SE) \vee (\exists SE.\ attribute)\}\}$$

$$< Logic7 > ::= \{TE \cap \{(\exists TEChild \subseteq SE) \vee (\exists TE.\ attribute)\}\} \leftarrow SE$$

$$< Logic8 > ::= \{TE \cap \{(\exists TEChild \subseteq SE) \vee (\exists TE.\ attribute)\}\} \leftarrow \{SE \cap \{(\exists SEChild \subseteq SE) \vee (\exists SE.\ attribute)\}\}$$

$$< Logic9 > ::= \{TE \cap \{\exists TEChild \subseteq TE \wedge (TEChild \geq 1)\}\} \leftarrow \{SE \cap \{\exists SE.\ attribute\}\}$$

$$< Logic10 > ::= \{TE \cap \{\exists TE.\ attribute \wedge (TE.\ attribute \geq 1)\}\} \leftarrow \{SE \cap \{\exists SEChild \subseteq SE\}\}$$

$$< Logic > ::= < LogicGM > < LogicCM >$$

The constructs $< CM >$ and $< LogicCM >$ are related to the customized mappings and are not covered in the scope of this paper; therefore its Backus–Naur Form is not presented. The rules in $< LogicCM >$ are the same as those in the $< LogicGM >$ construct. The detailed description of the ontology-alignment design pattern called PRM is elaborated in the following sections.

4. MBO design patterns

MBO utilizes Strategy Design Pattern and Mediator Pattern to incorporate an object-oriented design approach for agility and reusability of the system. It also used PRM to define a mapping representation format that can be used for easy expert verifications, format transformation and ontology translation purposes. Figure 2 illustrates a class diagram that shows MBO Strategy Design Pattern, MBO Mediator Pattern and PRMs as realization of the MBO in the SPHeRe system. MBO Strategy and Mediator design patterns explain the implementation view of the system design, while PRMs describe MBO ontology patterns as representation of the alignments. We have adopted the concept of Strategy and Mediator design patterns from the object-oriented design community and propose PRM in this research by interrelating them for extendible, flexible and agile system.

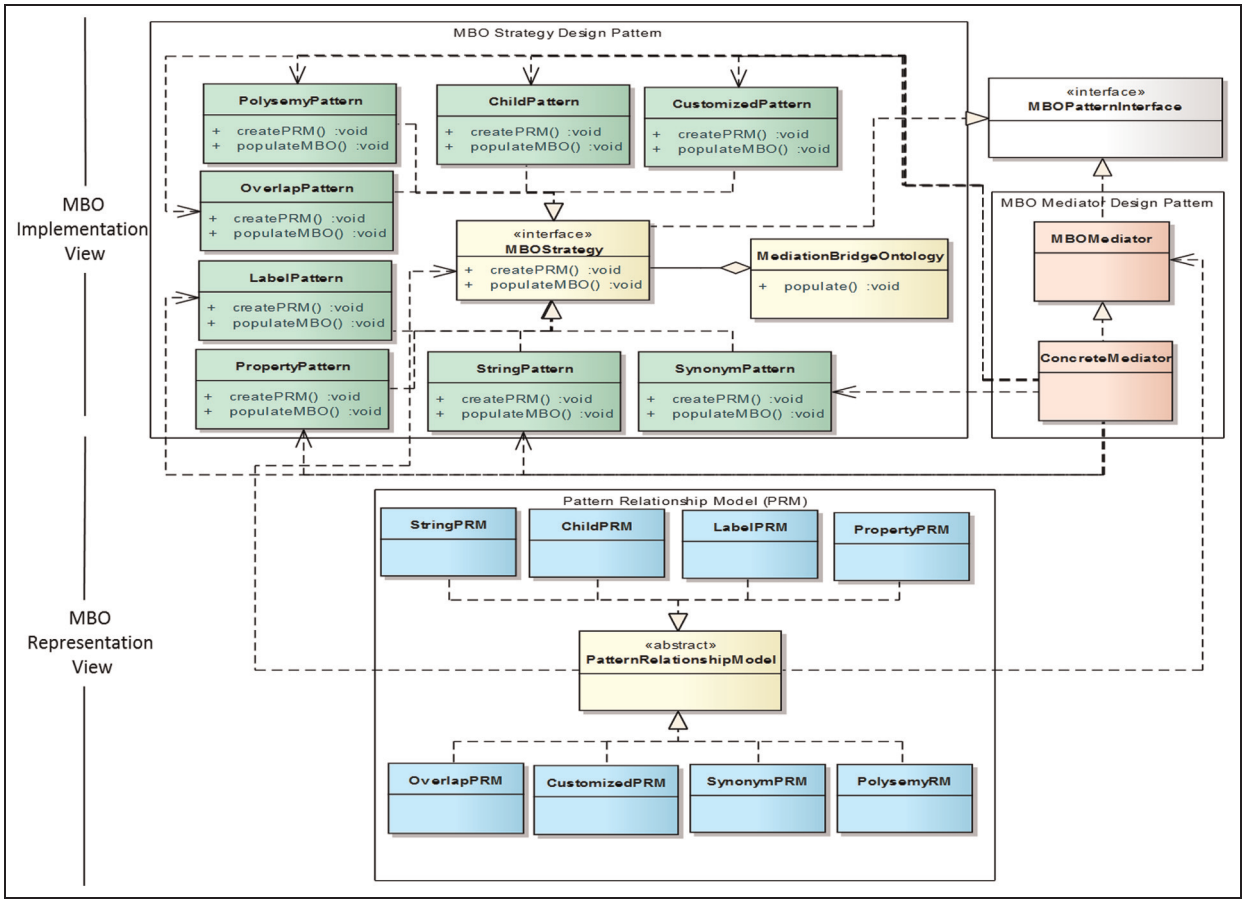


Figure 2. MBO design pattern-oriented implementation and representation views.

4.1. MBO implementation view

4.1.1. MBO Strategy Design Pattern.

Motivation: MBO is based on classes that differ only in their behaviour, therefore algorithms need to be isolated to provide the ability to select different algorithms at runtime.

Intent: Define a family of algorithms, encapsulate each one, and make them interchangeable. *MBOStrategy* lets the algorithm vary independently of clients that use it.

Applicability:

MBOStrategy – an interface that defines the behaviour of a *MediationBridgeOntology*.

Concrete Strategies: *ChildPattern*, *PropertyPattern*, *StringPattern*, *SynonymPattern*, *PolysemyPattern*, *OverlapPattern* and *LabelPattern*; each of these pattern classes calls a specific PRM for execution and then populates that information in the *MediationBridgeOntology*.

MediationBridgeOntology – this class is the context class that gets alignments information from each pattern and stores it in specified format.

4.1.2. MBO Mediation Design Pattern.

Motivation: MBO also provides classes that can use the services of other classes; therefore mediation is necessary between classes for reusability purposes.

Intent: Define an interface for communicating with related objects for understanding interdependencies among them. *MBOMediator* provides that interface to other objects for communicating with related objects.

Applicability:

MBOMediator – an interface class used for communicating with other objects in well-defined and complex ways.

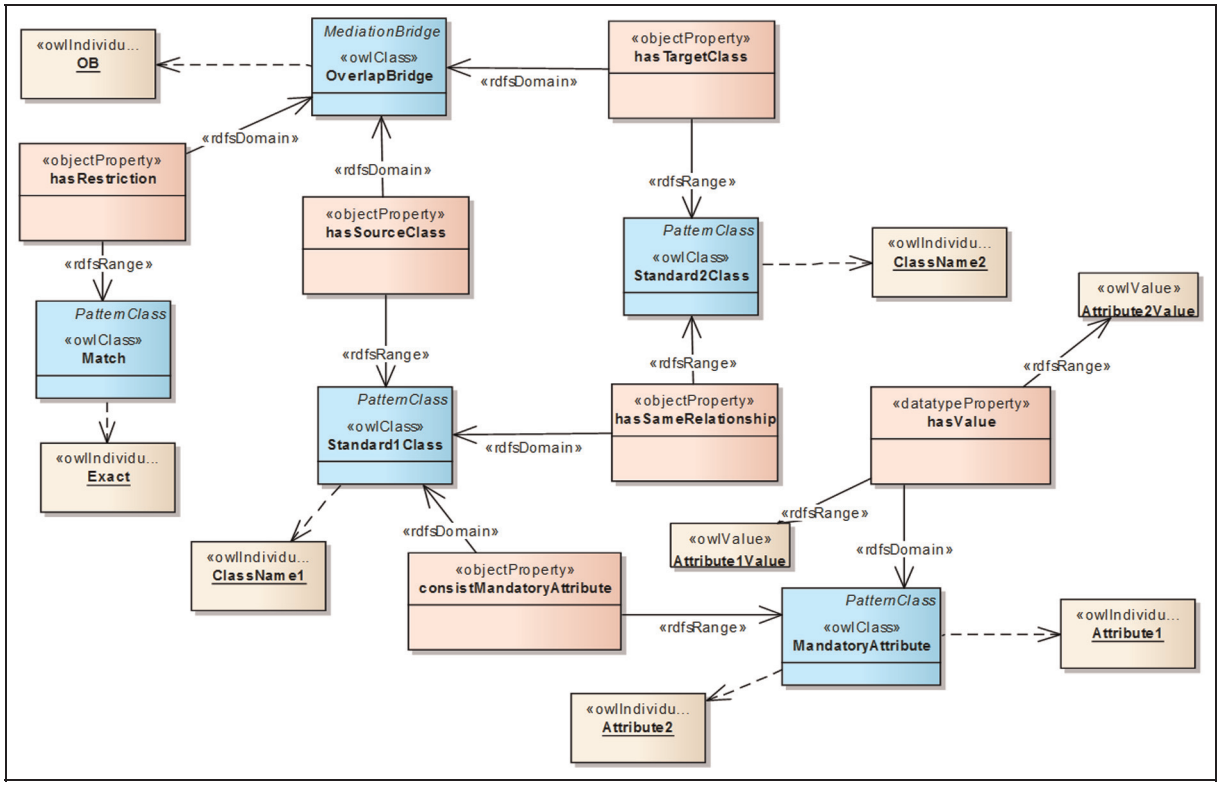


Figure 3. Overlap PRM [30].

ConcreteMediator – this class keeps reference of all the colleague objects and is used to transfer the messages between colleague classes such as *ChildPattern*, *PropertyPattern*, *StringPattern*, *SynonymPattern*, *PolysemyPattern*, *OverlapPattern* and *LabelPattern*.

4.2. MBO representation view (PRM)

Each pattern class in the Strategy Design Pattern uses a particular PRM class, for example, *StringPattern* class invokes *StringPRM* class for execution. All of the PRM classes are derived from the *PatternRelationshipModel* abstract class. Medical ontologies are used for matching purposes and performing experiments; therefore, medical standards are used as scenarios for understanding these PRMs. These PRM realizations are shown with Virtual Medical Record (VMR) and HL7 Clinical Document Architecture (CDA) standards ontologies. Both of the standards are based on the HL7 Reference Information Model [27] that is the root of all the information models and consists of backbone classes, and their specialization and structural attributes for further defining the roles of the classes. HL7 CDA follows a CDA Refined Message Information Model [28] that contains information about document creation and manipulation. VMR is a data model for representing clinical data relevant to CDS by recording patient demographics and clinical history data [29]. The generic pattern structure followed by its realization in VMR and CDA standard ontologies is described in this section. Some of these PRMs (OverlapPRM contents are also included in our paper [30]; we only changed the structure of the text based on the design pattern template) are explained in object-oriented design template below.

4.2.1. Overlap PRM.

Motivation: OverlapPRM deals with the type of alignment patterns where the source ontology concept with its mandatory attributes and values is mapped with the target ontology concept.

Intent: Define a mechanism to transform source and target concepts by taking into account mandatory attributes as well. The mapping representation targets *Overlap Bridge* of MBO.

Implementation: The pattern for Overlap Bridge is shown in Figure 3. *OverlapBridge* class has relationship with *Standard1Class* and *Standard2Class* through *hasSourceClass* and *hasTargetClass* object properties. *OverlapBridge* class

```
<rdf:RDF
 xmlns:vmr="http://www.owl-ontologies.com/VMR.owl#"
 xmlns:cda="http://www.owl-ontologies.com/CDA.owl#">
 <owl:Ontology rdf:about="BridgeOntology"/>
 <!-- Defining Classes for OverlapPRM -->
  <owl:Class rdf:ID="OverlapBridge"/>
  <owl:Class rdf:ID="MandatoryAttribute"/>
  <owl:Class rdf:ID="Match"/>
  <owl:Class rdf:ID="Standard1Class"/>
  <owl:Class rdf:ID="Standard2Class"/>
 <!-- Object Properties -->
   <owl:ObjectProperty rdf:ID="consistMandatoryAttributes">
    <rdfs:domain rdf:resource="#Standard1Class"/>
    <rdfs:range rdf:resource="#MandatoryAttribute"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasRelationship">
    <rdfs:domain rdf:resource="#OverlapBridge"/>
    <rdfs:range rdf:resource="#Match"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasSameRelationship">
    <rdfs:domain rdf:resource="#Standard1Class"/>
    <rdfs:range rdf:resource="#Standard2Class"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasSourceClass">
    <rdfs:domain rdf:resource="#OverlapBridge"/>
    <rdfs:range rdf:resource="#Standard1Class"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasTargetClass">
      <rdfs:domain rdf:resource="#OverlapBridge"/>
      <rdfs:range rdf:resource="#Standard2Class"/>
   </owl:ObjectProperty>
 <!-- CDA Class with its mandatory attributes and values -->
   <Standard1Class rdf:ID="CDA_EntryRelationship">
    <hasSameRelationship rdf:resource=
    "#RelatedClinicalStatement"/>
    <consistMandatoryAttributes rdf:resource="#TypeCode"/>
    <consistMandatoryAttributes rdf:resource=
    "#ContextConductionInd"/>
   </Standard1Class>
   <owl:DatatypeProperty rdf:ID="hasValue">
    <rdfs:domain rdf:resource="#MandatoryAttribute"/>
    <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>
   <MandatoryAttributes rdf:ID="ContextConductionInd">
    <hasValue rdf:datatype="&xsd;string">true</hasValue>
   </MandatoryAttributes>
   <MandatoryAttributes rdf:ID="TypeCode">
    <hasValue rdf:datatype="&xsd;string">CAUS</hasValue>
   </MandatoryAttributes>
   <Match rdf:ID="Exact"/>
 <!-- VMR Class -->
   <Standard2Class rdf:ID="VMR_RelatedClinicalStatement"/>
 <!-- Overlap Bridge Relationship -->
   <OverlapBridge rdf:ID="OverlapBridgeInd">
    <hasSourceClass rdf:resource="#EntryRelationship"/>
    <hasRelationship rdf:resource="#Exact"/>
    <hasTargetClass rdf:resource="#RelatedClinicalStatement"/>
   </OverlapBridge>
</rdf:RDF>
```

Figure 4. Overlap PRM Example (CDA and VMR) [30].

is related to *Match* class using *hasRelationship* object property with individuals *Exact* or *Subsume*. There are cases in which the mandatory properties of both of the standards are exactly matched while in some cases the source concept has a subsumption relationship with the target concept. *Standard1Class* and *Standard2Class* are also related to each other using *hasSameRelationship* object property. *Standard1Class* consists of *MandatoryAttribute* connected by *consistMandatoryAttributes* object property and this *MandatoryAttribute* contains some values represented by *hasValue* data type property. The realization of this pattern is given in Figure 4.

Applicability: HL7 CDA consists of classes in the form of triplet 'class-attribute-value'. Therefore, transformation of concepts between VMR and CDA the mandatory attributes transformation is necessary for correct parsing of the document. Overlap PRM deals with such type of patterns where the source standard concept with its mandatory attributes and values is converted into target concept. In this type of pattern an ontology O_i consisting of class C_i with mandatory attributes MA_i having values V_i is mapped with class C_j of another ontology O_j.

We explain *OverlapPRM* with *EntryRelationship* concept of CDA standard and *RelatedClinicalStatement* concept of VMR standard as shown in Figure 4. The *EntryRelationship* class of HL7 CDA has mandatory attributes such as *typeCode* and *contextConductionInd* with values *CAUS* and *true*, respectively. This information is mapped with the *RelatedClinicalStatement* class of VMR; therefore translation of *RelatedClinicalStatement* class is performed with *EntryRelationship* class and its mandatory attributes and values.

4.2.2. Property PRM.

Motivation: PropertyPRM deals with the type of alignment patterns where the properties of the source ontology concept match with the properties of the target ontology concept.

Intent: Define a mechanism to compare properties of source and target concepts and represent them as alignment if a particular threshold is reached. This pattern reflects the mappings for *Property Based Structural Bridge (PBSB)*.

Implementation: Figure 5 shows the property match pattern for PBSB class in the MBO. Three main classes, *PropertyBasedStrcuturalBridge*, *MappedClass* and *Match*, are related to each other by object properties *hasParticipatingClass*, *hasProperty* and *hasPropertyRestriction*. Each individual of PBSB class is related with *MappedClass* individuals from different standards by *hasParticipatingClass* object property. Each individual of *MappedClass* consists of properties in the form of *OWL:Class* related by *hasProperty* object property. These properties should have an exact or subsumption relationship with each other. Therefore, PBSB class individual is related to any of

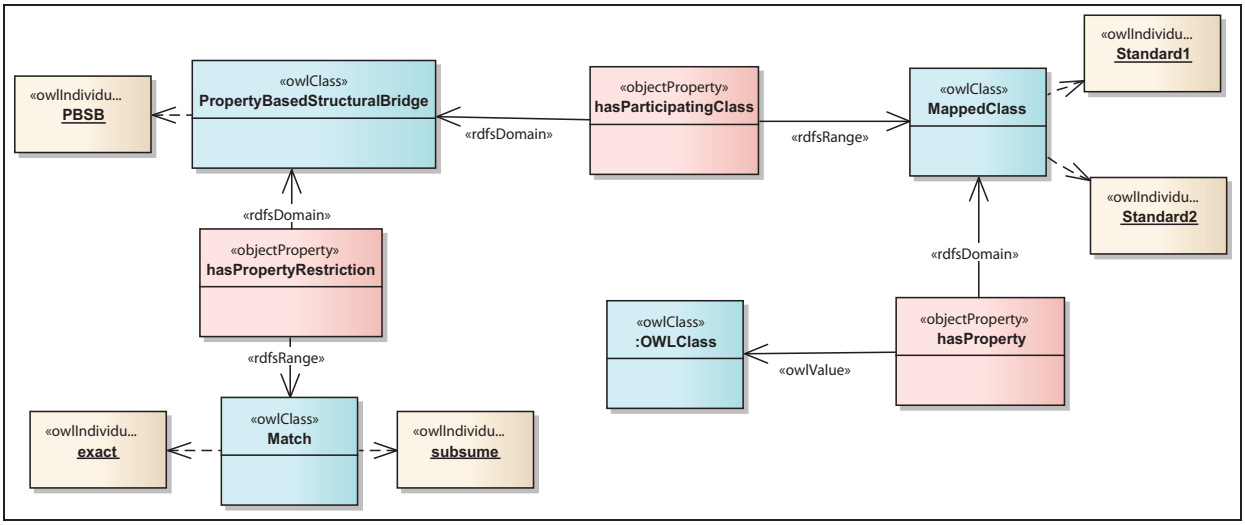


Figure 5. Property PRM.

```
<rdf:RDF
 xmlns:vmr="http://www.owl-ontologies.com/VMR.owl#"
 xmlns:cda="http://www.owl-ontologies.com/CDA.owl#">
 <owl:Ontology rdf:about="BridgeOntology"/>
 <!-- Defining Classes for Property Match Pattern -->
 <owl:Class rdf:ID="PBSB"/>
 <owl:Class rdf:ID="MappedClass"/>
 <owl:Class rdf:ID="Match"/>
 <!-- Properties of Observation Class in CDA -->
 <owl:Class rdf:about="&cda;Code"/>
 <owl:Class rdf:about="&cda;EffectiveTime"/>
 <owl:Class rdf:about="&cda;Value"/>
 <!-- Observation Class associated with its properties -->
 <MappedClass rdf:ID="CDA_Observation">
  <hasProperty rdf:resource="&cda;Code"/>
  <hasProperty rdf:resource="&cda;EffectiveTime"/>
  <hasProperty rdf:resource="&cda;Value"/>
 </MappedClass>
 <!-- Properties of ObservationResult class in VMR -->
 <owl:Class rdf:about="&vmr;ObservationEventTime"/>
 <owl:Class rdf:about="&vmr;ObservationFocus"/>
 <owl:Class rdf:about="&vmr;ObservationValue"/>
 <!-- ObservationResult class associated with its properties -->
 <MappedClass rdf:ID="VMR_ObservationResult">
  <hasProperty rdf:resource="&vmr;ObservationFocus"/>
  <hasProperty rdf:resource="&vmr;ObservationEventTime"/>
  <hasProperty rdf:resource="&vmr;ObservationValue"/>
 </MappedClass>
 <!-- Indiviual of Match class -->
 <Match rdf:ID="exact"/>
 <!-- Indiviual of PBSB class -->
 <PBSB rdf:ID="PBSB_INS_CDA_VMR">
  <hasPropertyRestricvtion rdf:resource="#exact"/>
  <hasParticipatingClass rdf:resource="#CDA_Observation"/>
  <hasParticipatingClass rdf:resource="#VMR_ObservationResult"/>
 </PBSB>
 <!-- Relationship between PBSB and MappedClass -->
 <owl:ObjectProperty rdf:ID="hasParticipatingClass">
  <rdfs:domain rdf:resource="#PBSB"/>
  <rdfs:range rdf:resource="#MappedClass"/>
 </owl:ObjectProperty>
 <!-- Relationship between MappedClass and OWL:Class -->
 <owl:ObjectProperty rdf:ID="hasProperty">
  <rdfs:domain rdf:resource="#MappedClass"/>
  <rdfs:range rdf:resource="&owl;Class"/>
 </owl:ObjectProperty>
 <!-- Relationship between PBSB and Match class -->
 <owl:ObjectProperty rdf:ID="hasPropertyRestricvtion">
  <rdfs:domain rdf:resource="#PBSB"/>
  <rdfs:range rdf:resource="#Match"/>
 </owl:ObjectProperty>
</rdf:RDF>
```

Figure 6. Example: *Observation* (CDA) and *ObservationResult* (VMR) property match pattern.

the *Match* class individuals using *hasPropertyRestriction* object property. This information identifies the nature of the relationship between the matched classes.

Applicability: An instantiation example for *PropertyPRM* is described in Figure 6. The *Observation* class of CDA standard is equivalent to the *ObservationResult* class of VMR standard based on their matching properties using *PropertyPRM*. The *Observation* class has *Code*, *EffectiveTime* and *Value* as its properties and the *ObservationResult* class has *ObservationFocus*, *ObservationEventTime*, *ObservationValue* properties. Observation's class property *Code* is related to the *ObservationFocus* property of *ObservationResult* class using *LabelPRM* and categorized under *Label Bridge*. In the same way, *EffectiveTime* and *Value* properties of *Observation* class are related to *ObservationEventTime* and *ObservationValue* properties of VMR class, respectively. *SynonymPRM*, which categorizes mapping information under *Synonym Bridge*, is used for *EffectiveTime* and *ObservationEventTime* properties, while *StringPRM* is used for *Value* and *ObservationValue* matching by categorizing it under *String Matching Bridge*.

4.3. Parameters for analysing design patterns quality

The proposed system utilizes the pattern design approach by integrating object-oriented design patterns with our proposed ontology-alignment design patterns. This delivers a solution for satisfying th functional, non-functional and alignment representation requirements of an ontology-matching system. Hsueh et al. [31] provide the motivation for adopting part of their object-oriented design pattern quality measure and use their tuple as $< I_F, I_N, Q >$:

- I_F: Functional Requirement Intent defines the functionality of the design pattern. For example, CBSB and PBSB bridge algorithms have the intention to match source and target ontology concepts based on their children and property matching, respectively.
- I_N: Non-functional Requirement Intent describes the level of attainment of quality attributes. For example, extendibility and reusability in the MBO case.
- Q: Quality Focus explains the quality focus between I_F and I_N.

We are using Strategy and Mediator design patterns that offer reusability and extendibility metrics to our system and provide assistance to ontology-alignment PRM with the the tuple as $< S,T,A,EC,MV >$:

- S: Source Concept that belongs to the matching source ontology.
- T: Target Concept that belongs to the matching target ontology.
- A: Attribute is supported by the evaluation criteria for matching and is divided into simple or composite attributes. A simple attribute performs matching with single evaluation criteria while composite includes multiple evaluation criteria combines on a single platform.
- EC: Evaluation Criteria defines the purpose of the alignment generation between concepts. Each bridge algorithm has particular evaluation criteria to achieve objective.
- MV: Matching Value decides about the fulfilment of the evaluation criteria.

The quality focus of the proposed system is on decrease coupling and increase polymorphism to achieve extendibility and reusability, described in detail in Section 6.4.

5. Methodology

Our proposed MBO is part of the ontology-matching system we developed called SPHeRe [9]. The SPHeRe system is based on different bridge algorithms that are represented in a mapping representation format provided by the MBO. Accuracy and performance are the two factors that help in achieving the goals, and these are accomplished by the *Matcher Library* and *Parallel Matching Framework* of the SPHeRe working model as shown in Figure 7. The definitions of the concepts used in the proposed architecture are described in Table 3.

Table 3. Concepts and definitions

Concept	Definition
SPHeRe Execution Control	It manages the communication with external and internal entities. It is responsible for ontology loading (source and target ontologies) and providing information about the execution of bridge algorithms in the Matcher Library to Parallel Matching Framework for parallel execution of the algorithms.
Parallel Matching Framework	It support parallel execution of matching bridge algorithms over multicore and multinode computational resources. The performance of the system is handled by this custom high-performance computing framework.
Matcher Library	It consists of the bridge algorithms that are invoked for performing the matching tasks. Each bridge algorithm generates mappings that are stored and represented in the MBO.
Distributor	It is responsible for the division of matching jobs over parallel hardware depending upon their computational ability.
Parallel Hardware Interface	It is used by *Distributor* to exploit the multiple cores available over commodity hardware for parallelism.
Aggregator	It accumulates the respective results of all matching jobs from all computing nodes after the completion of parallel matching.

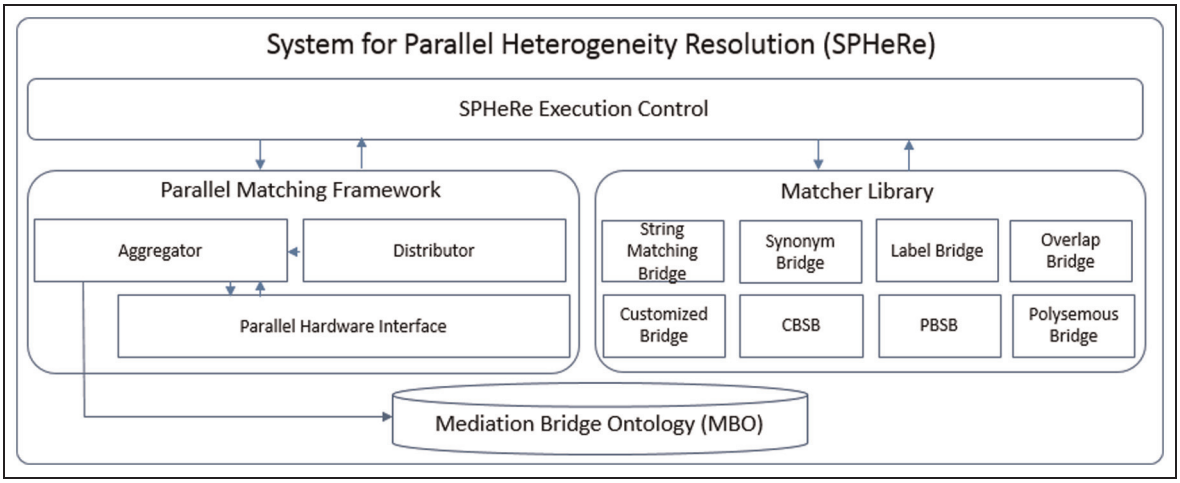


Figure 7. SPHeRe working model.

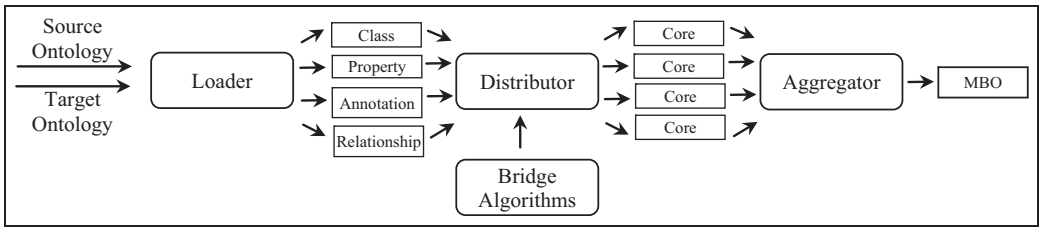


Figure 8. Process workflow

SPHeRe Execution Control manages the communication with external and internal entities. It is responsible for ontology loading and providing information about execution of bridge algorithms in the *Matcher Library* to *Parallel Matching Framework* for parallel execution of the algorithms. *Matcher Library* consists of bridge algorithms such as *String Matching Bridge*, *Synonym Bridge*, *Label Bridge*, *Overlap Bridge*, *Customized Bridge*, *CBSB*, *PBSB* and *Polysemous Bridge*.

Ontology matching being a computationally intensive problem requires adequate computational resources for effective resolution in acceptable time. To generate mappings with performance in perspective, we have implemented a custom high-performance framework, *Parallel Matching Framework*, to support parallel execution of matching algorithms over multicore and multinode computational resources. To accomplish parallel matching we have implemented two core components, that is, *Distributor* and *Aggregator*. *Distributor* is responsible for the division of matching jobs over parallel hardware depending upon their computational ability. The *Parallel Hardware Interface* is used by *Distributor* to exploit the multiple cores available over commodity hardware for parallelism. After the completion of parallel matching, the *Aggregator* component accumulates the respective results of all matching jobs from all computing nodes. This accumulated result is formalized by *Mediation Bridge Ontology*, to be further utilized as an alignment. The generated bridge ontology also persists in the repository for future utilization.

The process of SPHeRe working model is described in Figure 8. Initially, source and target ontologies are loaded for the matching process. Both of the ontologies are parsed based on the ontology constructs such as classes, properties, annotations and relationships. The distributor access the primary algorithms initially includes the String Matching, Label and CBSB bridge algorithms as some of their attributes are common. Based on these, ontology constructs are accessed and assigned to the cores for processing. Each core is assigned a specific task to perform in parallel for a particular bridge algorithm and the output is provided to the Aggregator to generate the MBO. In the same way, other algorithms are executed for the generation of mappings and their storage in MBO. Further details of this process working in a parallel environment are available in Amin et al. [9].

6. Evaluation

Existing ontology-matching systems mainly focus on the accuracy of mappings and lack assessment of the external quality factors from the measurement of the internal design properties. We evaluate our proposed system with COF, NOP and RoC metrics by comparing it with existing systems, FALCON and LogMap. We selected FALCON and LogMap for comparison with the proposed system because of factors such as participation in OAEI for several years, corresponding publications availability to understand their approach thoroughly, source code availability (to understand the design and implementation of the system) and also complete system availability (to run ontology-matching tests for observing the output). These systems class diagrams were generated from their source code using Intellij Idea tool,[5] which supports a wide array of refactoring for Java, cross language refactoring and other advanced features [32]. We used a Quality Model approach [10] to quantitatively assess the external factors such as extendibility and reusability as measures of software maintainability.

6.1. Coupling Factor

Coupling Factor is a metric to determine dependencies between the classes. Therefore, the formula to calculate COF is given in equation (1):

$$\mathrm{COF} = \frac{df}{tc^2 - tc} \tag{1}$$

where df = total dependency factor and tc = total number of classes.

The SPHeRe system is based on the MBO using object-oriented and ontology design patterns. Therefore, COF value of SPHeRe is less compared with FALCON and LogMap systems. Figure 2 shows the df and tc of the proposed system and the $\mathrm{COF}_{\mathrm{SPHeRe}}$ is calculated as shown in equation (2):

$$\mathrm{COF}_{\mathrm{SPHeRe}} = \frac{9}{12^2 - 12} = 0.068 \tag{2}$$

We compared our system with FALCON ontology-matching system and used its Matcher package to calculate the COF of its different subpackages as shown in Figure 9(a). We observed that FALCON has high coupling as compared with the proposed system. The class diagram of FALCON system's Package PBM is shown in Figure 9(b) and equation (3) calculates its COF value as 0.127, which is very high compared with the proposed system:

$$\mathrm{COF}_{\mathrm{FALCON}_{\mathrm{PackagePBM}}} = \frac{14}{11^2 - 11} = 0.127 \tag{3}$$

The LogMap system overall class diagram consists of approximately 26 packages and classes having too great a dependency on each other, resulting in highly coupled system. We selected two packages (Stemming and Reasoning) for comparison with the proposed system. These package class diagrams are shown in Figure 10. Figure 10(a and b) illustrates the class diagrams of LogMap system's Stemming and Reasoning packages respectively. The stemming package has more COF as compared with the proposed system while the Reasoning package has a lower COF value as shown in equations (4) and (5), respectively:

$$\mathrm{COF}_{\mathrm{LogMap}_{\mathrm{PackageStemming}}} = \frac{20}{14^2 - 14} = 0.11 \tag{4}$$

$$\mathrm{COF}_{\mathrm{LogMap}_{\mathrm{PackageReasoning}}} = \frac{13}{17^2 - 17} = 0.047 \tag{5}$$

6.2. Number of polymorphic methods

The number of polymorphic methods in a class diagram determines the value for polymorphism. Therefore, in Figure 2, it can be observed that *populateMBO()* is the polymorphic method that returns the *MBOStrategy* instance. Therefore, the NOP in a class diagram is the level of polymorphism, which is 7 in the proposed system, as shown in equation (6). This suggests that the system has more extendibility by implementing only the *populateMBO()* polymorphic method.

$$\mathrm{NOP}_{\mathrm{SPHeRe}} = 7 \tag{6}$$

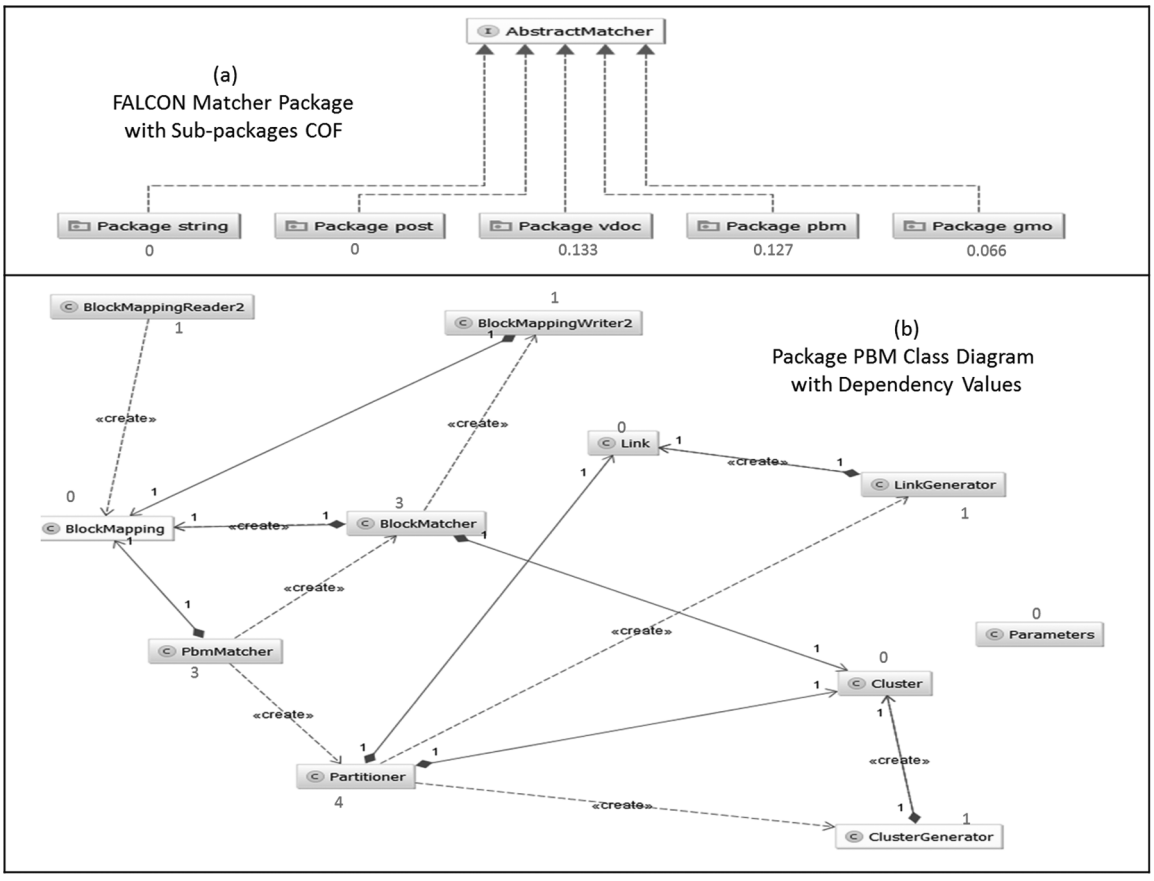


Figure 9. FALCON packages and coupling factor.

The increase in composition and association of a class diagram results in high coupling and less polymorphism. The FALCON class diagram shows more composition and association relationships whereas the proposed system contains more polymorphic methods in the class diagram. Figure 8(a) shows the extended relationship to the *AbstractMatcher* class, which suggests that there may be a polymorphic method in the class diagram of FALCON system's Package PBM, shown in Figure 8(b). Therefore, the maximum polymorphism value for FALCON system is 1, as shown in equation (7), which is less compared with the proposed system. A new bridge algorithm must implement the *populateMBO()* polymorphic method, thus increasing the polymorphism value. The LogMap system two packages polymorphism value is 5, as shown in equation (8), which is also less than the proposed system:

$$\mathrm{NOP}_{\mathrm{FALCON}} = 1 \tag{7}$$

$$\mathrm{NOP}_{\mathrm{LogMap}} = 5 \tag{8}$$

6.3. Rate of change

The key factor for a successful ontology-matching system is flexibility and extendibility based on new requirements. As new techniques and methodologies continuously evolve in the ontology-matching domain, measurement of RoC based on COF becomes necessary for evaluating the extendibility of the system. Therefore, RoC can be measured by equation (9), based on changes in the COF owing to addition of new classes and dependencies:

$$\mathrm{ROC} = \Delta\mathrm{COF} \tag{9}$$

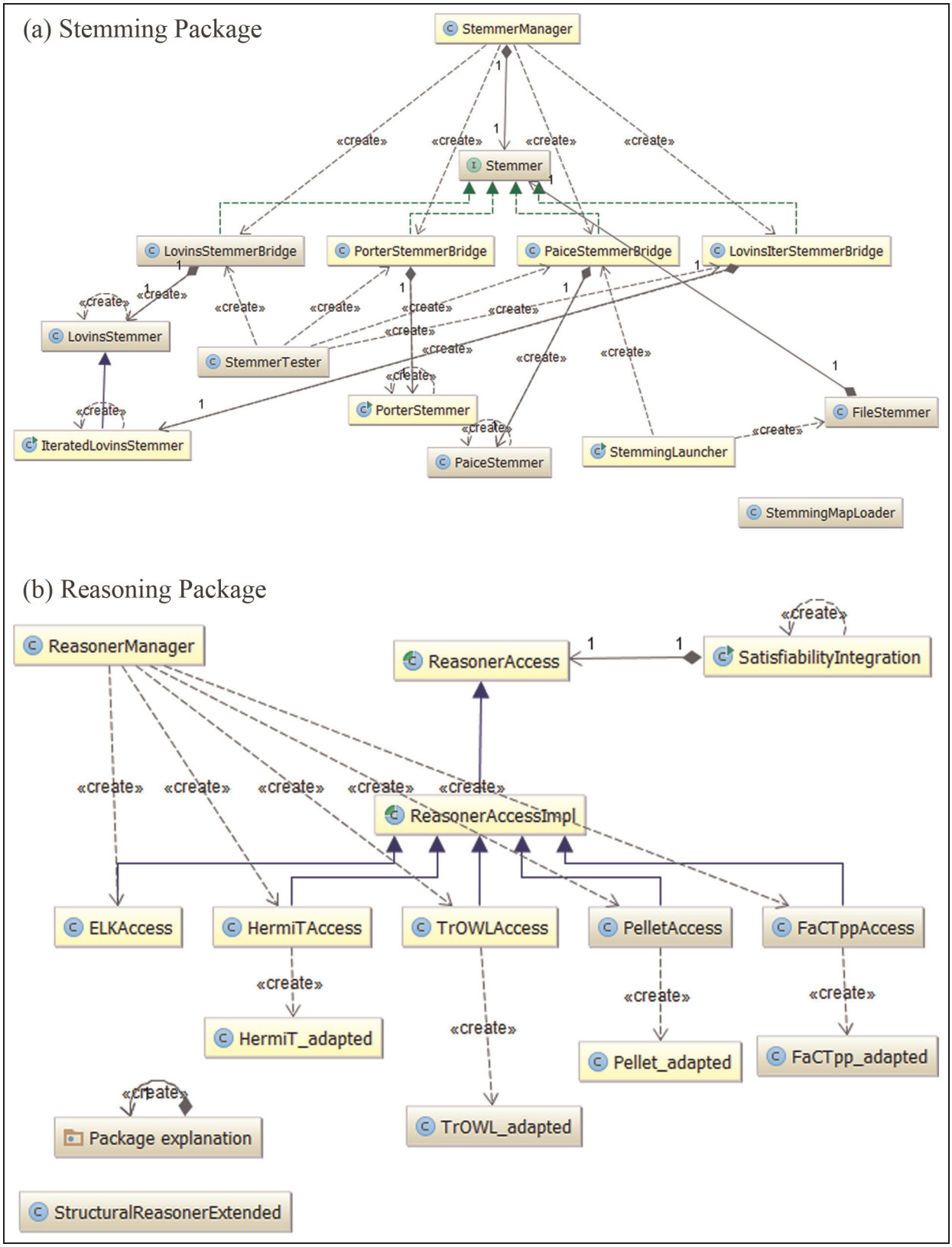


Figure 10. LogMap Class Diagrams: (a) Stemming Package and (b) Reasoning Package.

For testing the rate of change, we introduced unidirectional dependency of $+1$ in *df* and *tc*, so equations (2) and (3) are transformed to equations (10) and (11), respectively. In the same way LogMap's equations (4) and (5) are transformed to equations (12) and (13) respectively:

$$\mathrm{COF}_{\mathrm{SPHeRe}'} = \frac{10}{13^2 - 13} = 0.064 \tag{10}$$

$$\mathrm{COF}_{\mathrm{FALCON}'_{\mathrm{PackagePBM}}} = \frac{15}{12^2 - 12} = 0.114 \tag{11}$$

$$\mathrm{COF}_{\mathrm{LogMap}'_{\mathrm{PackageStemming}}} = \frac{21}{15^2 - 15} = 0.1 \tag{12}$$

$$\mathrm{COF}_{\mathrm{LogMap}'_{\mathrm{PackageReasoning}}} = \frac{14}{18^2 - 18} = 0.045 \tag{13}$$

The proposed system's RoC is considerably less than those for the FALCON and LogMap systems, which shows the extendibility and reusability features of our system and easy adaptation of new changes. Equations (14) and (15) show that the proposed system has the better capacity to accommodate any changes in the system design as compared with the FALCON system. LogMap's Stemming package has higher RoC while the Reasoning package has less RoC value as compared with the proposed system's RoC value. The RoC values for these packages are shown in equations (16) and (17):

$$\Delta\mathrm{COF}_{\mathrm{SPHeRe}} = \mathrm{COF}_{\mathrm{SPHeRe}} - \mathrm{COF}_{\mathrm{SPHeRe}'} = 0.068 - 0.064 = 0.004 \tag{14}$$

$$\Delta\mathrm{COF}_{\mathrm{FALCON}_{\mathrm{PackagePBM}}} = \mathrm{COF}_{\mathrm{FALCON}_{\mathrm{PackagePBM}}} - \mathrm{COF}_{\mathrm{FALCON}'_{\mathrm{PackagePBM}}} = 0.127 - 0.114 = 0.013 \tag{15}$$

$$\Delta\mathrm{COF}_{\mathrm{Logmap}_{\mathrm{PackageStemming}}} = \mathrm{COF}_{\mathrm{Logmap}_{\mathrm{PackageStemming}}} - \mathrm{COF}_{\mathrm{Logmap}'_{\mathrm{PackageStemming}}} = 0.11 - 0.1 = 0.01 \tag{16}$$

$$\Delta\mathrm{COF}_{\mathrm{Logmap}_{\mathrm{PackageReasoning}}} = \mathrm{COF}_{\mathrm{Logmap}_{\mathrm{PackageReasoning}}} - \mathrm{COF}_{\mathrm{Logmap}'_{\mathrm{PackageReasoning}}} = 0.047 - 0.045 = 0.002 \tag{17}$$

6.4. Discussion

Extendibility and reusability are the two main metrics for evaluation of the proposed system. These are discussed in relation to polymorphism and coupling of the proposed system measured in the previous subsections.

6.4.1. Extendibility. Extendibility is one of the evaluation metrics of the proposed system. A new bridge algorithm can easily be accommodated in the system design with low coupling, high polymorphism and lower rate of change as explained in previous section. This is achieved by using strategy design pattern with the PRMs. The new bridge algorithm only needs to implement the interface. We consider as a scenario that a new bridge is introduced that is based on instance-based matching, called the Instance Matching Bridge. *InstancePRM* is connected to the *PRM* in the MBO representation view that deals with actual representation of the alignment. A class *InstancePattern* will implement the *MBOStrategy* interface class and provide its reference information to the *ConcreteMediator* class. Therefore, its tuple metrics information is as follows:

- I_F: An algorithm to match source and target concepts based on instance comparison.
- I_N: *InstancePRM* and *InstancePattern* classes to be added in the class diagram to support extendibility. This algorithm resolves specific problems and only needs to implement an interface.
- Q: $< polymorphism, increased >$.
- S: Source Concept that belongs to the matching source ontology.
- T: Target Concept that belongs to the matching target ontology.
- A: Instances of source and target concepts.
- EC: Specific number of instance matches that source and target concepts are similar. A threshold value n should be achieved by the number of instances matched.
- MV: A value between 0 and 1 that is based on instances matched.

6.4.2. Reusability. A new bridge algorithm can be added to the system that can utilize existing bridge algorithms. Mediation between the new and existing bridges is performed using mediator design pattern and PRMs. For example, a

new bridge called Hyponym Bridge is introduced that uses CBSB and PBSB together to find matching concepts. *HyponymPRM* is connected to *PRM* in the MBO representation view, and the *HyponymPattern* class is also introduced to implement the *MBOStrategy* interface class and provide reference to the *ConcreteMediator* class. Tuple information is as follows:

- I_F: An algorithm to match source and target concepts based on existing CBSB and PBSB algorithms.
- I_N: *HyponymPRM* and *HyponymPattern* classes to be added in the class diagram for reusability.
- Q: $< coupling, decrease >$.
- S: Source Concept that belongs to the matching source ontology.
- T: Target Concept that belongs to the matching target ontology.
- A: Children and properties match of the matching concepts.
- EC: A specific number of children and properties match for source and target concepts match.
- MV: A value between 0 and 1 that is based on CBSB and PBSB results match.

These metrics enable easy integration of new bridge algorithms into the system that prolongs the system lifetime. State of the art matching techniques and new methodologies can be plugged-and-played into the proposed system, without disturbing the design of the system.

7. Conclusion and future work

Expressiveness in formal representation of alignments and the use of object-oriented and ontology-alignment design patterns prolongs the duration of use of ontology-matching systems. The proposed MBO approach uses Strategy and Mediator object-oriented design patterns with ontology-alignment design patterns, PRM, to support the extendibility and reusability aspects of the SPHeRe system. Evolution in matching techniques or the introduction of new bridge algorithms is made convenient by the proposed approach, and therefore is suitable for adoption by the ontology-matching community.

The effectiveness of the alignments stored in the MBO can be measured by evaluating a case study for transformation process between two ontologies of the same domain. Our objective is to match two medical standard-based ontologies with SPHeRe, store the alignments in the MBO and finally use those alignments for transformation from one medical standard ontology to another related to the same domain.

Acknowledgements

This work was supported by a post-doctoral fellowship grant from the Kyung Hee University Korea in 2011 (KHU-20110219).

Funding

This work was supported by a post-doctoral fellowship grant from the Kyung Hee University Korea in 2011 (KHU-20110219).

Notes

1. http://oaei.ontologymatching.org/
2. http://www.oodesign.com/strategy-pattern.html
3. http://www.oodesign.com/mediator-pattern.html
4. http://en.wikipedia.org/wiki/Backus%E2%80%93Naur_Form
5. http://www.jetbrains.com/idea/

References

[1] Shvaiko P and Euzenat J. Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering* 2013; 25: 158–176.

[2] Euzenat J, Meilicke C, Stuckenschmidt H, Shvaiko P and Cassia T. Ontology alignment evaluation initiative: Six years of experience. *Journal on Data Semantics* 2011; XV: 158–192.

[3] Vlissides J, Helm R, Johnson R and Gamma E. *Design patterns: Elements of reusable object-oriented software*. Reading: Addison-Wesley, 1995; 49:120.

[4] Gangemi A and Presutti V. Ontology design patterns. In: *Handbook on ontologies*. Berlin: Springer, 2009, pp. 221–243.

[5] Mortensen JM, Horridge M, Musen MA and Noy NF. Applications of ontology design patterns in biomedical ontologies. In: *AMIA annual symposium proceedings*. American Medical Informatics Association, 2012, p. 643.
[6] Presutti V and Gangemi A. Content ontology design patterns as practical building blocks for web ontologies. *Conceptual Modeling-ER*, 2008; 128–141.
[7] Scharffe F, Zamazal O and Fensel D. Ontology alignment design patterns. *Knowledge and Information Systems* 2012; 1–28.
[8] Scharffe F. Correspondence patterns representation. Doctoral dissertation, University of Innsbruck, 2009.
[9] Amin MB, Batool R, Khan WA, Lee S and Huh EN. SPHeRe: A performance initiative towards ontology matching. *The Journal of Supercomputing* 2013; 1–28.
[10] Bansiya J and Davis CG. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering* 2002; 28(1): 4–17.
[11] Xu P, Wang Y and Liu B. A differentor-based adaptive ontology-matching approach. *Journal of Information Science* 2012; 38(5): 459–475.
[12] Xie J, Liu F and Guan SU. Tree-structure based ontology integration. *Journal of Information Science* 2011; 37(6): 594–613.
[13] Hu W and Qu Y. Falcon AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web* 2008; 6(3): 237–239.
[14] Cruz IF, Sunna W, Makar N and Bathal S. A visual tool for ontology alignment to enable geospatial interoperability. *Journal of Visual Languages and Computing* 2007; 18(3): 230–254.
[15] Cruz IF, Antonelli FP and Stroe C. AgreementMaker: Efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment* 2009; 2(2): 1586–1589.
[16] Faria D, Pesquita C, Santos E, Palmonari M, Cruz IF and Couto FM. The agreementmakerlight ontology matching system. In: *On the move to meaningful Internet systems: OTM 2013 conferences*. Berlin: Springer, 2013, pp. 527–541.
[17] Kirsten T, Gross A, Hartung M and Rahm E. GOMMA: A component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of Biomedical Semantics* 2011; 2: 6.
[18] Jiménez-Ruiz E and Grau BC. Logmap: Logic-based and scalable ontology matching. In: *The Semantic Web–ISWC*. Berlin: Springer, 2011, pp. 273–288.
[19] Ngo D, Bellahsene Z and Coletta R. A flexible system for ontology matching. In: *IS Olympics: Information systems in a diverse world*. Berlin: Springer, 2012, pp. 79–94.
[20] Ngo D and Bellahsene Z. YAM++ : A multi-strategy based approach for ontology matching task. In: *Knowledge engineering and knowledge management*. Berlin: Springer, 2012, pp. 421–425.
[21] Šváb-Zamazal O, Svátek V and Iannone L. Pattern-based ontology transformation service exploiting OPPL and OWL-API. In: *Knowledge engineering and management by the masses*. Berlin: Springer, 2010, pp. 105–119.
[22] Thomas H, O'Sullivan D and Brennan R. Evaluation of ontology mapping representation. In: *Proceedings of the workshop on matching and meaning*, 2009, pp. 64–68.
[23] Xu B, Wang P, Lu J, Li Y and Kang D. Bridge ontology and its role in semantic annotation. In: *International conference on cyberworlds*. New York: IEEE, 2004, pp. 329–334.
[24] Wang P, Xu B, Lu J, Li Y and Kang D. Theory and semi-automatic generation of bridge ontology in multi-ontologies environment. In: *On the move to meaningful internet systems 2004: OTM 2004 workshops*, 2004, pp. 763–767.
[25] Euzenat J and Shvaiko P. *Ontology matching*, vol. 18. Berlin: Springer, 2007
[26] Navarro G. A guided tour to approximate string matching. *ACM Computing Surveys* 2001; 33(1): 31–88.
[27] Beeler GW, Huff S, Rishel W, Shakir AM, Walker M, Mead C and Schadow G. Message development framework. Technical report, Version 3.3, Health Level Seven Inc., December 1999.
[28] Dolin RH, Alschuler L, Boyer S, Beebe C, Behlen FM, Biron PV and Shvo AS. Hl7 clinical document architecture, release 2. Technical report, HL7 V3 Normative Edition, Health Level Seven Inc., 2011.
[29] Health Level Seven Inc. Domain analysis model: Virtual medical record for clinical decision support (vmr-cds), release 1. Technical report, January 2012, http://www.hl7.org/v3ballot2012jan/html/dams/uvvmr/uvvmr.html (accessed August 2012).
[30] Khan WA, Khattak AM, Hussain M, Amin MB, Afzal M, Nugent C and Lee S. An adaptive semantic based mediation system for data interoperability among health information systems. *Journal of Medical Systems* 2014; 38(8): 1–18.
[31] Hsueh NL, Chu PH and Chu W. A quantitative approach for evaluating the quality of design patterns. *Journal of Systems and Software* 2008; 81(8): 1430–1439.
[32] Jemerov D. Implementing refactorings in IntelliJ IDEA. In: *Proceedings of the 2nd workshop on refactoring tools*. New York: ACM, 2008, p. 13.