The MOUSE Approach: Mapping Ontologies Using UML For System Engineers

Prof. Seung-Hwa Chung[1,*], Dr. Wei Tai[2], Prof. Declan O'Sullivan[2], Dr. Aidan Boran[3]

CSE Department, Bennett University, Greater Noida, India

School of Computer Science and Statistics, 2Trinity College Dublin, Dublin, Ireland.

Bell Labs Ireland, Alcatel-Lucent, Dublin, Ireland

*seung-hwa.chung@bennett.edu.in

Abstract:

To address the problem of semantic heterogeneity, there has been a large body of research directed toward the study of semantic mapping technologies. Although various semantic mapping technologies have been investigated, facilitating the process for domain experts to perform a semantic data integration task is still not easy. This is because one is required not only to possess domain expertise but also to have a good understanding of knowledge engineering. This paper proposes an approach that automatically transforms an abstract semantic mapping syntax into a concrete executable mapping syntax, we call this approach MOUSE (**M**apping **O**ntologies using **U**ML for **S**ystem **E**ngineers). In order to evaluate MOUSE, an implementation of this approach for a semantic data integration use case has been developed (called SDI, Semantic Data Integration). The aim is to enable domain experts, particularly system engineers, to undertake mappings using a technology that they are familiar with (UML), while ensuring the created mappings are accurate and the approach is easy to use. The proposed UML-based abstract mapping syntax is evaluated through usability experiments conducted in a lab environment by participants who have skills equivalent to real life system engineers using the SDI tool. Results from the evaluations show that the participants could correctly undertake the semantic data integration task using the MOUSE approach while maintaining accuracy and usability (in terms of ease of use).

Keywords: Knowledge Engineering; Ontology Mapping; Semantic Data Integration; UML; System Engineers

I. INTRODUCTION

To enable the encoding of semantics with the data, well-known technologies are RDF (Resource Description Framework) [1] and OWL (Web Ontology Language) [2]. These technologies formally represent the meaning involved in information. For example, ontology can describe concepts, relationships between things, and categories of things. These embedded semantics with the data offer significant advantages such as reasoning over data and dealing with heterogeneous data sources [3][4]. However, the ontology that represents the knowledge within a certain domain may be developed within different engineering processes resulting in heterogeneous ontologies in both conceptualization and terminology. Conceptual heterogeneity can occur due to the natural human diversity involved in ontology development of a particular domain or due to the differences between the ways in modelling and interpreting entities that depend on differing viewpoints and different portions of the domain [5][6]. Terminological heterogeneity can occur when mismatches relating to the naming process of the ontology entities use different words to name the same entity (synonymy), the same word is used to name different entities (polysemy), words from different languages (multilingualism) and syntactic variations of the same word (different acceptable spellings, abbreviations, use of optional prefixes or suffixes, and so on) [7][8].

Ontology heterogeneity typically requires mappings to exchange information in a semantically sound manner [9]. There have been investigations into ontology mapping technologies to resolve the ontology heterogeneity issues that are often encountered during the integration of ontology data from various sources. The existing ontology mapping approaches [10] usually require mapping practitioners to have a considerable amount of

expertise in knowledge engineering in order to perform the mapping process, and ontology mappings are often performed by knowledge engineers. Hence in most of the current situations, a mapping practitioner also needs to be a knowledge engineer. In this research, it is argued that performing a semantic data integration task by system engineers such as telecommunications system engineers that are considered domain experts is more realistic because of the complexity in designing semantic mappings for non-trivial cases in the system. However, since the creation of ontology mappings requires considerable effort [11] and considerable amount of expertise in knowledge engineering [12], it is understandable that not all the system engineers are able to perform semantic data integration tasks. The research presented in this paper focuses on supporting those system engineers who are expected to have insufficient ontological knowledge or lack knowledge engineering experience.

Considering that system engineers may have little background in ontology techniques, the semantic mapping conceptualization needs to be abstract, meaning that the syntax needs to be more natural to manipulate than a concrete syntax [13]. A concrete mapping syntax means that the syntax can be directly executable in a system to perform integration [14].

Thus in this research an approach is proposed - MOUSE (Mapping Ontologies using UML for System Engineers) - that will support system engineers expressing the mappings that they require, and automatically transform these abstract mapping expressions into executable expressions of the mappings within the system.

II. Related Work

This section presents research on ontology mapping relationships in order to categorize the mapping types that define the scope of the mapping relationships supported by the abstract mapping syntax. This categorization is done in order to consider what extent will the proposed MOUSE approach allow the creation of mappings using an abstract syntax. This section also reviews existing abstract and concrete mapping syntaxes that can describe ontology mapping relationships in order to consider the transformation of the abstract mappings into concrete executable mappings. There is a certain ambiguity between 'Abstract Syntax' and 'Concrete Syntax' that also needs to be defined. This research defines these two categorical syntaxes as following: (1) abstract syntax is the format that is independent of particular representation, and it is more convenient and natural to manipulate than concrete syntax; and (2) concrete syntax is the format that can be derived from the abstract syntax, is ready to be used in a system, or is the target of a specific machine representation or encoding.

The evaluation criteria used in this research are their capacity to express the ontology mapping relationships discussed in the categorization of the mapping types and their usability to generate and manipulate mapping relationships in the first place by human perspective. At the end of this section, each discussed abstract and concrete mapping syntax is summarized in a table that describes each mapping syntax's capability to support the mapping relationships and usability to generate mapping relationships from a human perspective.

A. Ontology mapping categorization

There are various kinds of patterns for classification in the ontology engineering field [15]. This section specifically reviews the work in Francois Scharffe and Dieter Fensel's research [16], which is a renowned research study in the ontology mapping field. The research study is well-known because the researchers surveyed patterns in ontology mapping relationships and published detailed correspondence patterns that are at the top level of abstraction of the ontology alignment representation stack.

Scharffe's research of correspondence patterns defines mapping relationships in six generic patterns: (1) Equivalent correspondence pattern: the pattern usually used to solve a terminological mismatch; (2) Subsumption correspondence pattern: the pattern typically solves a granularity mismatch; (3) Conditional pattern: the pattern requires a restriction to narrow down the scope of an entity in an ontology to match the scope of an entity in the another ontology, e.g. a restriction on the scope of a class based on 1. the occurrence,

2. the value or 3. the type of an attribute; (4) Transformation pattern: the pattern requires a transformation of a property value to fit the corresponding property in the another ontology, e.g. the transformation about 1. data type conversion, 2. unit transformation and 3. currency conversion; and (5) Union and (6) Intersection patterns: these patterns are used to relate entities modelled at a different granularity.

In the next sections, these patterns will be used in the discussion of the concrete and abstract mapping representations to illustrate their relative capabilities and limitations.

B. ***Concrete mapping representations***

This section reviews 3 concrete syntaxes for their ability to represent mappings that are directly executable by the system: OWL, SWRL and SPARQL. Each of these are representative of one of three different approaches of integration that are popular: Axiomatic, Rule and Query [17].

1) ***Using OWL for Semantic Mapping***

Web Ontology Language (OWL) [18] is a semantic markup language for publishing and sharing ontologies on the World Wide Web. This language has XML elements such as equivalent Class, equivalent Property that can be used to describe mappings. If the application is ontologically based, this syntax is generally the first consideration as a concrete syntax to perform the integration task. However, it has some limitations. OWL does not permit arithmetic operations that might be required for mappings, and using OWL for the mappings mixes definitions of the concepts and mapping information. This representation by itself does not support data value transformations that require arithmetic operations.

2) ***SWRL***

Semantic Web Rule Language (SWRL) [19] is a rule language to express the rules in an ontology, not only concept definitions. This syntax is based on a combination of OWL and RuleML [20] that is a sublanguage of the Rule Markup Language. The syntax includes a high-level abstract syntax for Horn-like rules, and uses vocabularies: XSD, OWLX, RuleML, SWRLX, SWRLB. These vocabularies offer good expressiveness to describe mappings that may require expressing data range constraints or unit transformations. However, this syntax is difficult to generate for its rigid syntax often require domain experts to have intensive training on writing rules.

3) ***SPARQL***

SPARQL Protocol and RDF Query Language (SPARQL) [21] is a query language for RDF [22], that is a directed and labeled graph data format for representing information. SPARQL can transform semantic data at query time, as a CONSTRUCT statement can generate the desired RDF graph based on the results of the query. This syntax offers good expressiveness to describe mappings that may require expressing data range constraints or unit transformations. However, this syntax is not easy to generate for a user who is unfamiliar with semantic web technologies.

All these concrete semantic mapping representations are suited to be used by a mapping practitioner who has a considerable amount of expertise in knowledge engineering.

C. ***Abstract mapping representations***

This section reviews the following abstract syntaxes: Alignment Format, EDOAL, C-OWL, SSE, SQWRL, RIF, all of which are able to describe semantic mapping relationships at an abstract level. UML is then discussed.

1) ***Alignment Format***

Alignment Format [23] was developed to express a set of pairs of mapping elements from source and target ontologies in an XML format. This syntax is composed of three different classes. The first one is the Alignment class. This class describes the general mapping information with the properties: xml, level, type, onto1, onto2, map. The second one is the Cell class. This class describes the detail mapping information between two entities with the properties: entity1, entity2, measure, relation. The last one is the Relation class. The customized mapping relationships can be expressed in this class as a string. This syntax uses simple vocabularies and offers good extendibility, but does not offer much expressiveness. For example, data range and unit transformation related mappings is not possible to be described using this syntax. This representation is well known for its practical use in the Ontology Alignment Evaluation Initiative (OAEI) Campaign [24] as it is the formal mapping document format for this initiative. However, it focuses primarily on one to one mappings.

2) ***EDOAL***

Expressive and Declarative Ontology Alignment Language (EDOAL) [25] extends the Alignment Format by extending the expressiveness. Particularly, it is designed for the representation of complex mappings. This syntax enables more precise relationships between entities, more than equivalence or subsumption relationships, by using the operators: and, or, not, compose, inverse, transitive, reflexive, symmetric. These enable more mapping expressiveness than Alignment Format such as describing data range related mapping. However, this syntax can support only limited unit transformation related mappings. It is not possible to express the complex unit transformation that might require the combination of multiple datatype property values for a mapping.

3) ***C-OWL***

Contextualized OWL (C-OWL) [26] has been developed to map contextualized ontologies (local models encoding the view of a group of people on a domain) to a shared ontology (a shared model encoding a common view of different parties on some domains). C-OWL keeps mappings explicitly and extends the OWL syntax. It uses bridge rules to express the mapping. These rules have five different attributes, i.e. equiv, onto, into, compat, and incompat, each of which describes a particular relationship, respectively: equivalent, less specific (subsumes), more specific (subsumedBy), compatible, and disjoint. This syntax represents the mappings in a simple and explicit way between source and target ontologies. However, it is limited in its ability to express data range constraints or unit transformation mappings.

4) ***SSE***

SPARQL Syntax Expressions (SSE) [27] has been developed in the JENA Apache incubator project to abstractly express SPARQL algebra expressions. This syntax offers good expressiveness to describe mappings that may require expression of data range constraints or unit transformation, as it supports SPARQL expressivity. However, even though it expresses SPARQL syntax with a more abstract approach, this syntax still follows almost the same structure as SPARQL, requiring knowledge engineering experience.

5) ***SQWRL***

Semantic Query-Enhanced Web Rule Language (SQWRL) [28] has been developed in the Stanford Center for Biomedical Information Research Lab in order to express rules abstractly and to query OWL ontologies. This syntax has been used in the Protégé tool to facilitate the creation and modification of the SWRL syntax. Its query structure is based on SWRL and it supports SWRL expressivity with more abstracted expressions. This syntax is relatively easier than SSE syntax to generate in the first place by human perspective.

6) ***RIF***

Rule Interchange Format (RIF) [29] has been developed in the W3C RIF Working Group to create a standard for exchanging rules among rule systems. This syntax offers good expressiveness to describe mappings that may

require expressing data range constraints or unit transformation, as it covers the capacity for expressivity of SWRL. However, this syntax is difficult to generate, as the structure and terms are complicated, and it uses many externally defined functions.

7) ***UML***

Unified Modeling Language (UML) is widely used in many areas to model data structure, application structure, application behavior, and business processes as a standardized and easy-to-understand means. It offers a graphical representation of entities and relationships, which is more readable to non-technical users than a text based expression. There has been research to describe ontology using UML, and as a result, ODM standard [30], which describes how ontological terms can be mapped to UML notations, has been published. However, there are limitations on describing mapping relationships using ODM. Firstly this ODM specification only supports OWL version 1 and it shares the limitations of OWL version 1, i.e. the expression of data range constraint (this expression is supported from OWL version 2 [31]), and secondly OWL does not support arithmetic expression. This research tried to cope with these lacks of expressing mapping relationships by using OCL standard with UML notations. OCL [32] is a formal language that enables UML constraints to be modelled in an unambiguous means. It avoids contradiction of the intended UML semantics by describing rules between UML notations. There has been a research on UML based semantic mapping representation initially developed an abstract mapping representation that combines ODM and OCL standards. ODM is used to represent ontology vocabularies in UML notations, and OCL is used to fill the mapping expressiveness gap of ODM. However, the usability of this proposed abstract mapping syntax was shown, through the feedback of an experiment on the tool that implements this proposed mapping syntax, to be unsatisfactory for non-ontology experts due to ODM standard uses customized stereotypes in UML to define terms in ontology and generating OCL was difficult task for general system engineers. It was discovered that, to be usable by system engineers, the syntax had to be more abstract.

D. ***Summary Tables***

This section summarizes each discussed mapping syntax's capability to support mapping types and their relative usability to generate mapping relationships in the first place by human perspective in the opinion of the author of this paper as shown in table 1 and table 2.

TABLE 1 SUMMARY TABLE OF THE SUPPORTED MAPPING TYPES FOR CONCRETE AND ABSTRACT MAPPING SYNTAXES

Concrete Syntax	Supported Mapping Types					
	E. Type	S. Type	C. Type	T. Type	U. Type	I. Type
OWL 2	X	X	X		X	X
SWRL	X	X	X	X	X	X
SPARQL	X	X	X	X	X	X

Abstract Syntax	Supported Mapping Types					
	E. Type	S. Type	C. Type	T. Type	U. Type	I. Type
Alignment Format	X	X				
EDOAL	X	X	X	X	X	X
C-OWL	X	X				
SSE	X	X	X	X	X	X
SQWRL	X	X	X	X	X	X
RIF	X	X	X	X	X	X
UML/ODM	X	X			X	X
UML/OCL	X	X	X	X	X	X

E. Type: Equivalent mapping type **C. Type:** Conditional mapping type **U. Type:** Union mapping type
S. Type: Subsumption mapping type **T. Type:** Transformation mapping type **I. Type:** Intersection mapping type

TABLE 2 SUMMARY TABLE OF THE USABILITY AND TYPE OF EXPRESSION FOR CONCRETE AND ABSTRACT MAPPING SYNTAXES

Concrete Syntax	Usability	Type of Expression
OWL 2	Difficult to Generate, Very Difficult to Update	Structured and Text-based Expression
SWRL	Difficult to Generate, Difficult to Update	Structured and Text-based Expression
SPARQL	Difficult to Generate, Easy to Update	Structured and Text-based Expression
Abstract Syntax	**Usability**	**Type of Expression**
Alignment Format	Very Easy to Generate, Very Easy to Update	Structured and Text-based Expression
EDOAL	Difficult to Generate, Difficult to Update	Structured and Text-based Expression
C-OWL	Very Easy to Generate, Very Easy to Update	Structured and Text-based Expression
SSE	Very Difficult to Generate, Easy to Update	Structured and Text-based Expression
SQWRL	Easy to Generate, Difficult to Update	Enumerated and Text-based Expression
RIF	Difficult to Generate, Easy to Update	Structured and Text-based Expression
UML/ODM	Very Easy to Generate, Easy to Update	Graphic-based Expression
UML/OCL	Difficult to Generate, Easy to Update	Graphic and Text (Structured) based Expression

Usability: Indication of how difficult to generate in the first place and to update (manipulate) by human perspective
- Level: Very Easy, Easy, Difficult, Very Difficult

Type of Expression: The expression is classified into four types.
1. Enumerated: Expressing relationship in line formation
2. Structured: Expressing relationship in structure formation
3. Text-based Expression: If the expression is represented based on text
4. Graphic-based Expression: If the expression is represented graphically such as diagram

UML [33] is widely used in many areas to model data structure, application structure, application behaviour, and business processes in a standardized and easy-to-understand manner [34]. It is the dominant language becoming the de facto standard for describing system applications, and system engineers are familiar with UML notations [35]. In addition, the UML based abstract semantic mapping representation offers a graphical representation of entities and relationships, which is more readable to non-technical users than a text based expression.

It was thus decided in our research to use UML and OCL-like syntax as the means to allow an engineer express mappings in an abstract manner. In this research, a more abstracted UML based semantic mapping representation is developed. This abstract representation uses no stereotyped UML notations for ontology vocabularies that require system engineers to know ontology specific knowledge, and uses the syntactic sugar that facilitates the use of OCL constrain.

III. MOUSE APPROACH

This research has developed a UML semantic mapping representation suitable for use by system engineers. A tool has been developed that implements the proposed UML semantic mapping representation. The tool, called SDI (Semantic Data Integration), automatically generates executable semantic mappings from the UML notations created by the user to express the desired mappings.

A. *The Core Mapping Types*

This section describes three core mapping types has derived from an industry use case: (1) Direct Mapping Type, (2) Data Range Mapping Type and (3) Unit Transformation Mapping Type that need to be captured correctly for the abstract syntax to be usable in a practical integration situation.

SDA (Semantic Data Access) is a research program initiated by Alcatel-Lucent Bell Labs Ireland [36]. This program aims to create a semantic data access plane within the telecommunications network. This access plane

will facilitate the modelling of, integration of, and reasoning of heterogeneous data sources and will support third-party enterprises accessing data in a unified approach for lifting rough refined raw semantic data to rich semantic data. The industry use case used in this research was the result of the research [37] conducted by Bell Labs Ireland about empirical analysis for network performance classification.

The use case is about the classification of network performance by semantic up-lift using real network ontology data collected from an Alcatel-Lucent femtocell [38] test bed. The performance management (PM) data is periodically collected and stored as an instance in raw-level ontology and then processed to generate key performance indicators (KPI) from the calculation over the PM data. The performance of the femtocell network is evaluated according to the KPIs and classified in femto-level ontology. This process requires semantic mappings to integrate a femtocell instance in raw-level ontology to femto-level ontology. For example, a femtocell is classified as 'HighHandoverFemto' in femto ontology if the generated KPI value of 'BSR_cluster_to_MTS_underlay_Intra_Frequency_Hard_Handover_Failure_Rate' is higher or equal to 90. The KPI values are generated by processing the PM data collected from the femtocell in the telecommunications network, i.e. 'BSR_cluster_to_MTS_underlay_Intra_Frequency_Hard_Handover_Failure_Rate' = (1 - ('VS_HHO_SuccBsrUmtsIntraFreq' value / 'VS_HHO_AttBsrUmtsIntraFreq' value)) * 100).

From the industry use case, this research has observed that the semantic mappings were performed only within the scope of the ontology class or the datatype property (it is understandable considering the industry use case is from telecommunications network performance data). Table 3 shows the mapping between the identified semantic integration situations and the defined three core mapping types.

TABLE 3 RELATIONSHIP BETWEEN THE SEMANTIC INTERGRATION SITUATIONS AND THE CORE MAPPING TYPES

Integration situations		Mapping relationships		Mapping types
Integration of ontology class instances into another ontology class without any constraint	=	Equivalent or subsumption mapping relationship	=	Direct mapping type
Integration of selected instances in one ontology class into another ontology class by the range constraint of the property value	=	Conditional mapping relationship that constraints the value of property (data range)	=	Data range mapping type
Integration of ontology class instances into another ontology class with the value transformation of the instance property	=	Transformation mapping relationship that transforms the value of property (unit transformation)	=	Unit transformation mapping type

These mapping types had more priority than others because, they were the only three semantic mapping types from the supplied industry use case in the telecommunications network performance domain. To satisfy the needs of the industry use case, a proposed abstract mapping syntax must express these core mapping types in order to be usable in a practical integration situation. In addition, the core mapping types identified from the industry use case were also observed in the ontology mapping categorization. Table 4 shows the relationship between these core mapping types and the correspondence patterns.

TABLE 4 RELATIONSHIP BETWEEN THE GENERIC MAPPING TYPES AND THE CORE MAPPING TYPES

Core Mapping Types	Link	Generic Patterns
Direct Mapping Type	= equivalent to	**Equivalent correspondence pattern**
		Subsumption correspondence pattern
Data Range Mapping Type	⊂ subset of	**Conditional pattern** e.g. a restriction on the scope of a class based on: - The occurrence of a property - **The value of a property (Data Range)** - The type of a property
Unit Transformation Mapping Type	⊂ subset of	**Transformation pattern** e.g. a transformation of a property value about: - Datatype Conversion - **Unit Transformation** - Currency Conversion
	void	**Union and intersection patterns**

It was determined that these core mapping types would be required to be captured correctly for the abstract syntax to be usable in a practical integration situation. Therefore, the initial focus was placed on these three mapping types in order to develop the SDI (Semantic Data Integration) tool to implement the proposed abstract mapping syntax; allowing it to be used in experiments to evaluate the proposed MOUSE approach.

B. *A walked through example*

This subsection describes the use of the proposed abstract mapping syntax explained in previous sections. To generalize the use of the core mapping types (not limited to the telecommunications domain), this research tried to apply these core mapping types in another domain, and the conference domain was selected because this domain is more generalized than the telecommunications domain (that normally requires deep understanding of the domain specific knowledge) and is generally understandable by more people with a system engineering background.

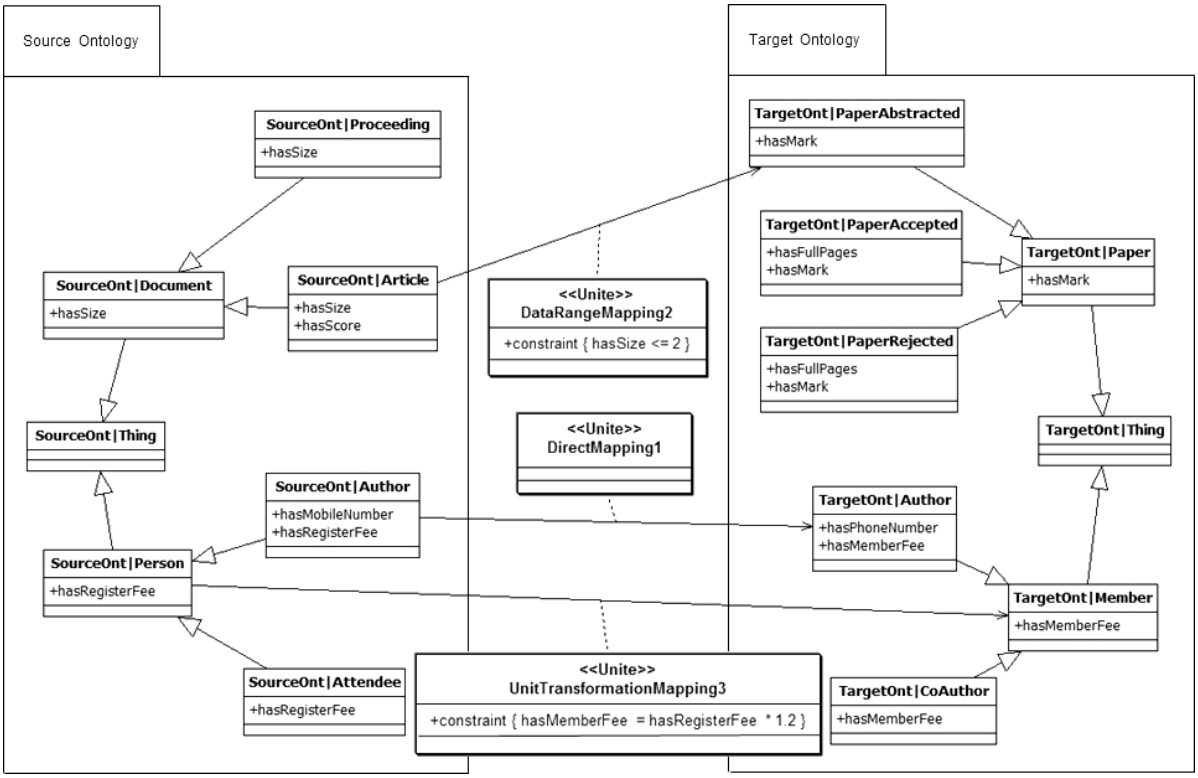

Figure 1 Example of the UML-based abstract mapping syntax

Fig. 1 shows two conference ontologies turned into UML notations, e.g. "SourceOnt|Article" class with 'hasSize' and 'hasScore' properties in the UML model represents the ontology class 'Article' with datatype property 'hasSize' and 'hasScore' in SourceOnt ontology. This figure also shows how the conference UML models can be mapped together using UML notations outlined earlier. In the figure, there are examples of applications for the three core mapping types: direct mapping, data range mapping and unit transformation mapping. For example, the "DataRangeMapping2" association class with the constraint in property represents the semantic mapping of data range mapping type.

1) Direct Mapping

UML::AssociationClass '<<Unite>> DirectMapping1' will be executed first among the three semantic mappings in the UML diagram. Author class (which has hasMobileNumber and hasRegisterFee properties) in the source ontology and Author class (which has hasPhoneNumber and hasMemberFee properties) in the target ontology are connected via a DirectMapping association class indicating that the Author class in the source ontology is mapped to the other Author class in the target ontology with the integration property corresponding to "Unite the source class (Author) individuals to the target class (Author)".

2) Data Range Mapping

UML::AssociationClass '<<Unite>> DataRangeMapping2' will be executed second among the three semantic mappings in the UML diagram. Article class (which has hasSize and hasScore properties) in the source ontology and PaperAbstracted class (which has hasMark property) in the target ontology are connected via a DataRangeMapping association class indicating that the Article class in the source ontology is mapped to the PaperAbstracted class in the target ontology with the integration property corresponding to "Unite the source ontology class (Article) individuals, that satisfies the data range constraint (hasSize <= 2), to the target ontology class (PaperAbstracted)". This means that a paper with less than or equal to 2 pages in the Article class corresponds to an abstract paper in the PaperAbstracted class.

3) Unit Transformation Mapping

UML::AssociationClass '<<Unite>> UnitTransformationMapping3' will be executed third among the three semantic mappings in the UML diagram. Person class (which has hasRegisterFee) in the source ontology and Member class (which has hasMemberFee) in the target ontology are connected via a UnitTransformationMapping association class indicating that the Person class in the ontology is mapped to the Member class in the target ontology with the integration property corresponding to "Unite the source ontology class (Person) individuals to the target ontology class (Member) with the new assigned data value (hasMemberFee = hasRegisterFee*1.2) for the datatype property (hasMemberFee) by the arithmetic transformation of the datatype properties (hasRegisterFee)". This means the registration fee is multiplied by 1.2 (e.g. because of the currency difference) paid by a person in the Person class corresponds to the amount of the member fee paid by a person in the Member class.

C. SDI Tool Development

This section describes the development of a tool that implements the proposed abstract mapping syntax and the automatic transformation process that constitutes the MOUSE approach. The tool, called SDI (Semantic Data Integration), allows a system engineer to express mappings using UML and then, automatically generates the corresponding executable semantic mappings. It also generates an integrated ontology as a computation result of executing the generated executable mappings on the source and target ontologies. This tool is designed to make it easier for system engineers to perform a semantic data integration task.

1) Conversion Process

The SDI tool enables the transformation of UML notations into the Rule Interchange Format (RIF) [39] and subsequently into SPARQL [21] queries which is a concrete mapping syntax and ready to be executable. In this research, RIF was chosen as the intermediate format because it has the potential to interoperate with other concrete mapping syntaxes. For example, there is an existing RIF syntax specification [40] for RDF and OWL Compatibility. This strategy potentially enables the transformation between the intermediate format and other concrete mapping syntaxes such as the ontology for axiom-based integration in future. SPARQL was chosen as a target concrete mapping syntax for Query-based integration approach. Using SPARQL has proven to be the most practical approach - in terms of integration process speed - among three different semantic data integration approaches [17], i.e. Axiom-based integration using ontology, Rule-based integration using SWRL and Query-based integration using SPARQL. There is no loss of expressivity in using RIF as an intermediate syntax between UML notations and SPARQL queries for the core mapping types because the proposed UML notations are designed to support the core mapping types. RIF and SPARQL support direct, conditional and transformation mapping types that may require expressing data range constraints or data value transformations (arithmetic operation) as discussed in related research. Fig. 2 shows the overall SDI tool generation process transforming the UML notations into concrete executable mappings in SPARQL queries through the intermediate RIF syntax.

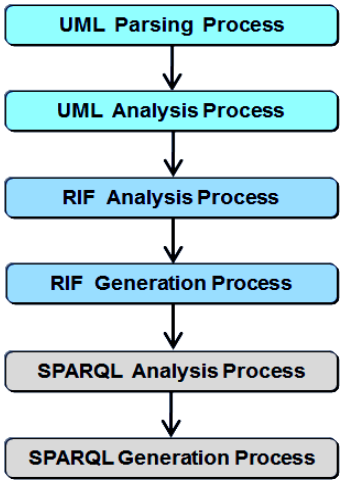

Figure 2 SDI tool generation process overview

2) UML to RIF Conversion

The UML notations (UML and UML constraint expressions) are interpreted into the RIF syntax. The following sections describe the translation method from UML notations into the RIF syntax for each core mapping type.

a) Direct Mapping Type

This section describes the translation method and an example from UML notations into the RIF syntax for the direct mapping type. Table 5 shows the RIF syntax according to UML notations with corresponding ontology syntax for the notation.

TABLE 5 CORRESPONDENCE BETWEEN UML NOTATION AND RIF SYNTAX FOR DIRECT MAPPING TYPE

UML Notation	OWL Syntax	RIF Syntax	Comment
Package::ownedMember Classifier	Ontology Document	Import with OWL RDF-Based entailment	Package name is the document name. Multiple documents can be imported.
First part of the classifier separated by a double colon ('::')	A prefix that represents the namespace of document	Prefix (<Name> < <Namespace> >)	<Name> is an string entity that represents <Namespace> in IRI/URI form.
UML::Class with the classifier 'Ont1::Class1' in the name field	<owl:Class rdf:about="Ont1:Class1"/>	?blankNode [rdf:type -> Ont1:Class1]	This type of syntax is called RIF frame formula of the RDF triple correspondent form. Each '?blankNode', 'rdf:type', and 'Ont1:Class1' corresponds to subject, predicate, and object in RDF triple.
<<Unite>> stereotyped UML::AssociationClass with the name 'DirectMapping'	<owl:Class rdf:about="#Class2"> <owl:equivalentClass rdf:resource="#Class1"/> </owl:Class>	?blankNode [rdf:type -> <UML::Class1>] :- ?blankNode [rdf:type -> <UML::Class2>]	"ϕ :- ψ" is a rule implication. Consequent (ϕ) is true, when the antecedent (ψ) is true.

Fig. 3 shows a translation example of UML notations into the RIF syntax according to the table 5. In this figure, bold and italic keywords in the RIF syntax are a template which means these keywords will not be altered by the UML notation changes in this mapping type. UML notations in Fig. 3 represent the mapping corresponding to "Integrate 'Layer3Switch' class individuals to 'Router' class".

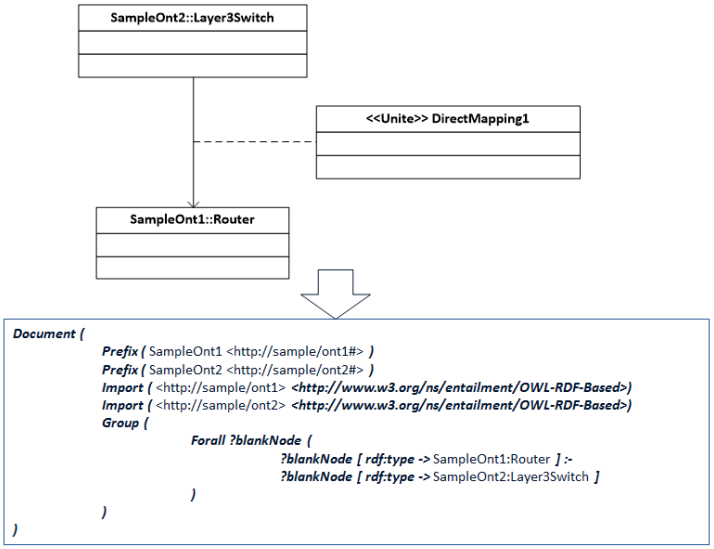

Figure 3 Translation example of UML notations to RIF syntax for direct mapping type

b) Data Range Mapping Type

This section describes the translation method and examples from UML notations into the RIF syntax for the data range mapping type. Table 6 shows the RIF syntax according to UML notations with corresponding ontology syntax for the notation.

TABLE 6 CORRESPONDENCE BETWEEN UML NOTATION AND RIF SYNTAX FOR DATA RANGE MAPPING TYPE

UML Notation	OWL Syntax	RIF Syntax	Comment
<<Unite>> stereotyped UML::AssociationClass with the name 'DataRangeMapping'		Prefix (pred <http://www.w3.org/2007/rif-builtin-predicate#>)	Prefix for RIF predicate built-ins for comparison operation
Comparison operator in UML::Constraint such as '<', '>', '<=', '>=', '==', or '<>'		External (pred: <RIF predicate built-in for comparison operation>)	<RIF predicate built-in for comparison operation> can be one of following: numeric-equal(), numeric-not-equal(), numeric-less-than(), numeric-less-than-or-equal(), numeric-greater-than(), or numeric-greater-than-or-equal(). [RIF Built-Ins]
Classifier used in UML::Constraint of the property 'Constraint' represents UML::Property of a certain UML::Class.	<owl:DatatypeProperty rdf:ID="A" />	?blankNode [A -> ?value]	?value can be used as a variable bounded in a RULE.
Classifier used in UML::Constraint of the property 'Constraint' represents UML::Property of a certain UML::Class.	<owl:DatatypeProperty rdf:ID="A" /> <owl:DatatypeProperty rdf:ID="B" />	?blankNode [A -> ?v1] ?blankNode [B -> ?v2]	Adding the RIF frame formula in FORMULA part of RULE within 'And' or 'Or' relation depending on the number of datatypeProperty variables. ?v1 and ?v2 can be used as a variable bounded in a RULE.
UML::Class at the association to represents a target class.	<rdf:Description rdf:ID="blankNode"> <rdf:type rdf:resource="#object" /> </rdf:Description>	?blankNode [rdf:type -> ?object]	This represents the constrained class instances, i.e. blankNode, are member of the ?object class.

Fig. 4 shows a translation example of UML notations into the RIF syntax according to the table 6. In the figure, bold and italic keywords in the RIF syntax are a template which means these keywords will not be altered by the UML notation changes in this mapping type. UML notations in Fig. 4 represent the mapping corresponding to "Integrate 'Femto' class individuals that are constrained by the data range (successRate1 > 90) of the datatype property 'successRate1' to 'HighHandoverFemto' class".

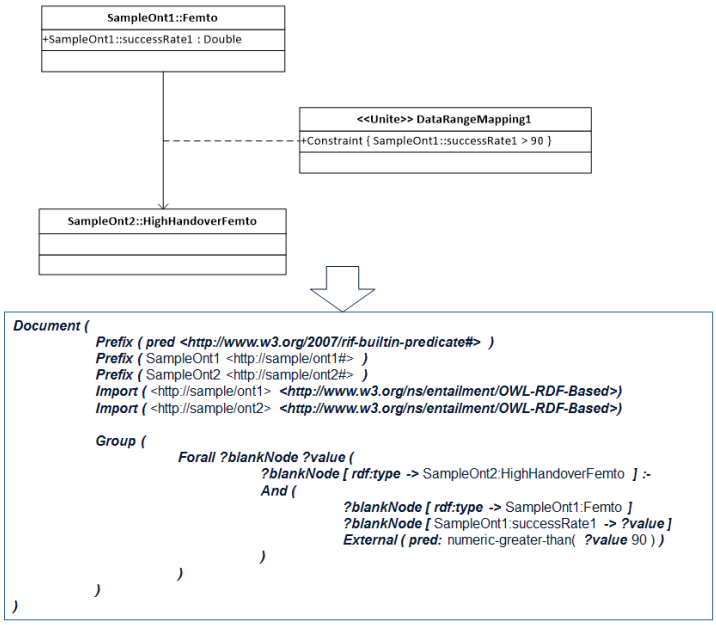

Figure 4 Translation example of UML notations to RIF syntax for data range mapping type

c) *Unit Transformation Mapping Type*

This section describes the translation method and examples from UML notations into the RIF syntax for the unit transformation mapping type. Table 7 shows the RIF syntax according to UML notations with corresponding ontology syntax for the notation.

TABLE 7 CORRESPONDENCE BETWEEN UML NOTATION AND RIF SYNTAX FOR UNIT TRANSFORMATION MAPPING TYPE

UML Notation	OWL Syntax	RIF Syntax	Comment
<<Unite>> stereotyped UML::AssociationClass with the name 'UnitTransformationMapping'		Prefix (func <http://www.w3.org/2007/rif-builtin-function#>)	Prefix for RIF functional built-ins for arithmetic operation
Arithmetic operator in UML::Constraint such as '*', '/', '+', or '-'		External (pred: <RIF functional built-in for mathematical operation>)	<RIF functional built-in for mathematical operation> can be one of following: numeric-add(), numeric-subtract(), numeric-multiply(), numeric-divide(). [RIF Built-Ins]
Classifier used in UML::Constraint of the property 'Constraint' represents UML::Property of a certain UML::Class.	<owl:DatatypeProperty rdf:ID="A" /> <owl:DatatypeProperty rdf:ID="B" />	?blankNode [A -> ?value1] ?blankNode [B -> ?value2]	?value1 and ?value2 can be used as a variable bounded in a RULE.
UML::Class at the association to represents a target class.	<rdf:Description rdf:ID="blankNode"> <rdf:type rdf:resource="#object" /> </rdf:Description>	?blankNode [rdf:type -> ?object]	This represents the constrained class instances, i.e. blankNode, are member of the ?object class.

Fig. 5 shows a translation example of UML notations into the RIF syntax according to the table 7. In the figure, bold and italic keywords in the RIF syntax are a template which means these keywords will not be altered by the UML notation changes in this mapping type. UML notations in Fig. 5 represent the mapping corresponding to "Integrate 'Femto' class individuals that have the datatype property 'FailureRate' to 'Device' class that has the datatype property 'FailureGauge' with the transformed value by the arithmetic operation (FailureGauge = FailureRate * 100)".

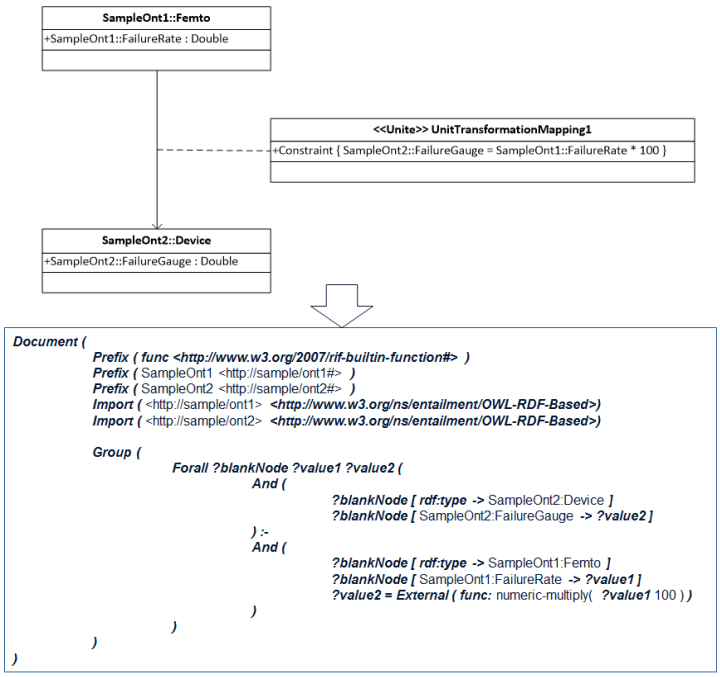

Figure 5 Translation example of UML notations to RIF syntax for unit transformation mapping type

3) RIF to SPARQL Conversion

The RIF syntax interpreted from the UML notations is translated into SPARQL queries. This section describes the translation method from the RIF syntax to SPARQL queries and examples for each core mapping type. Table 8 shows the SPARQL syntax based on the RIF syntax which is based on the UML notations.

TABLE 8 CORRESPONDENCE BETWEEN RIF SYNTAX AND SPARQL SYNTAX

RIF Syntax	SPARQL Syntax	Comment
Prefix	PREFIX	Prefix places in the beginning of RIF document.
Group	A query consists of CONSTRUCT and WHERE clauses.	RIF document can contain several groups, and each group represents one complete SPARQL query.
?blankNode	$instance	?blankNode is used in a RULE. RULE is the term used in RIF for a rule implication (ϕ :- ψ). Consequent (ϕ) is true, when the antecedent (ψ) is true.
rdf:type	a	rdf:type places in Frame used in RULE. Frame is the term used in RIF for "TERM '[' (TERM '->' TERM)* ']'", where TERM can be "constant, variable, list or (external) expression".
ATOMIC Frame	CONSTRUCT clause	ATOMIC in RIF means consequent part (ϕ) of the rule.
FORMULA	WHERE clause	FORMULA in RIF means antecedent part (ψ) of the rule.
RIF predicate built-in for comparison operation	FILTER	FILTER in SPARQL is used to express the constraint. In this case, comparison operation is expressed in this filter.
RIF functional built-in for arithmetic operation	BIND	BIND in SPARQL allows a value to be assigned to a variable from a graph pattern.

Fig. 6 shows a translation example of a RIF document to SPARQL query according to the table 8 for the direct mapping type. SPARQL query in Fig. 6 represents the mapping that all the instances in Layer3Switch class by WHERE clause will be an instance of Router class by CONSTRUCT clause.

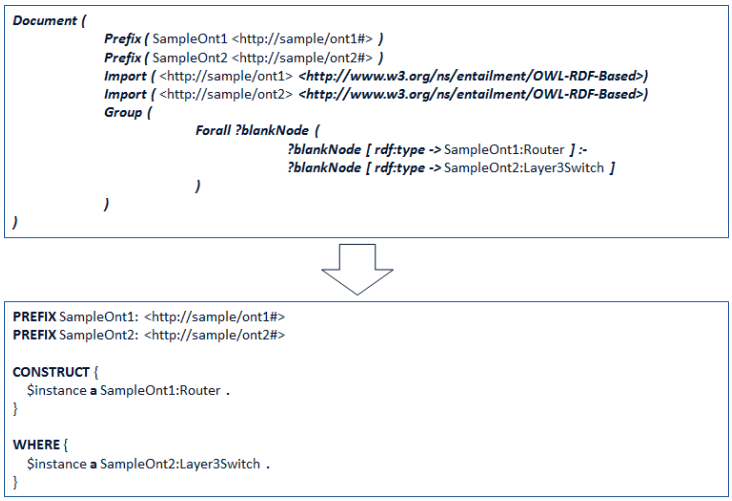

Figure 6 Translation example of RIF document to SPARQL query for direct mapping type

Fig. 7 shows a translation example of a RIF document to SPARQL query according to the table 8 for the data range mapping type. SPARQL query in Fig. 7 represents the mapping that the instances in Femto class, which the property successRate1 is constrained by FILTER (value of the property > 90), will be an instance of HighHandoverFemto class by CONSTRUCT clause.

```
Document (
        Prefix ( pred <http://www.w3.org/2007/rif-builtin-predicate#> )
        Prefix ( SampleOnt1 <http://sample/ont1#> )
        Prefix ( SampleOnt2 <http://sample/ont2#> )
        Import ( <http://sample/ont1> <http://www.w3.org/ns/entailment/OWL-RDF-Based>)
        Import ( <http://sample/ont2> <http://www.w3.org/ns/entailment/OWL-RDF-Based>)

        Group (
                Forall ?blankNode ?value (
                        ?blankNode [ rdf:type -> SampleOnt2:HighHandoverFemto ] :-
                        And (
                                ?blankNode [ rdf:type -> SampleOnt1:Femto ]
                                ?blankNode [ SampleOnt1:successRate1 -> ?value ]
                                External ( pred: numeric-greater-than( ?value 90 ) )
                        )
                )
        )
)
```

```
PREFIX SampleOnt1: <http://sample/ont1#>
PREFIX SampleOnt2: <http://sample/ont2#>

CONSTRUCT {
        $ instance a SampleOnt2:HighHandoverFemto .
}

WHERE {
        $ instance a SampleOnt1:Femto .
        $ instance SampleOnt1:successRate1 $value .
        FILTER ( $value > 90 )
}
```

Figure 7 Translation example of RIF document to SPARQL query for data range mapping type

Fig. 8 shows a translation example of a RIF document to SPARQL query according to the table 8 for the unit transformation mapping type. SPARQL query in Fig. 8 represents the mapping that the instances in Femto class, which the value of property FailureRate is assigned to a variable with arithmetic operation by BIND (value of the property * 100), will be an instance of Device class by CONSTRUCT clause with a new value of property FailureGauge from the variable in BIND clause.

```
Document (
        Prefix ( func <http://www.w3.org/2007/rif-builtin-function#> )
        Prefix ( SampleOnt1 <http://sample/ont1#> )
        Prefix ( SampleOnt2 <http://sample/ont2#> )
        Import ( <http://sample/ont1> <http://www.w3.org/ns/entailment/OWL-RDF-Based>)
        Import ( <http://sample/ont2> <http://www.w3.org/ns/entailment/OWL-RDF-Based>)

        Group (
                Forall ?blankNode ?value1 ?value2 (
                        And (
                                ?blankNode [ rdf:type -> SampleOnt2:Device ]
                                ?blankNode [ SampleOnt2:FailureGauge -> ?value2 ]
                        ) :-
                        And (
                                ?blankNode [ rdf:type -> SampleOnt1:Femto ]
                                ?blankNode [ SampleOnt1:FailureRate -> ?value1 ]
                                ?value2 = External ( func: numeric-multiply( ?value1 100 ) )
                        )
                )
        )
)
```

```
PREFIX SampleOnt1: <http://sample/ont1#>
PREFIX SampleOnt2: <http://sample/ont2#>

CONSTRUCT {
        $instance a SampleOnt2:Device .
        $instance SampleOnt2:FailureGauge $value2 .
}
WHERE {
        $instance a SampleOnt1:Femto .
        $instance SampleOnt1:FailureRate $value1 .
        BIND ( ( $value1 * 100 ) AS $value2 )
}
```

Figure 8 Translation example of RIF document to SPARQL query for unit transformation mapping type

Fig. 9 shows an example of semantic mapping documents generated automatically by the tool from UML notations that describe semantic mappings. The SDI tool automates the transformation of UML notations into RIF document and subsequently into SPARQL executable queries for the integration.

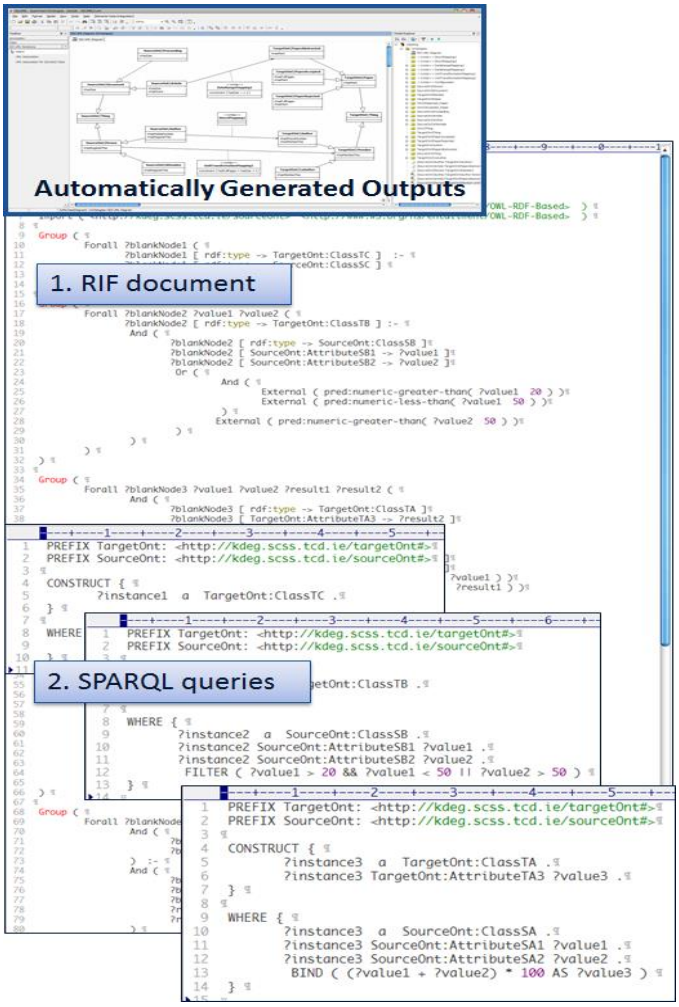

Figure 9 Example of Auto-generated SDI Tool Outputs

IV. EXPERIMENT

A usability experiment was conducted on the SDI Tool to investigate whether the proposed UML based abstract semantic mapping representation is usable (in terms of accuracy and ease of use) for system engineers who expected to be not familiar with ontology or not trained for the ontology mapping.

A. *Method*

The experiment used mapping design worksheets to capture the participant's mapping intention. This mapping design worksheet shows source and target ontologies using UML notations and the participant described the mappings using natural language and draw lines if necessary. The SDI tool was used to draw mappings using UML notations from the mapping design worksheets, and generated RIF document and SPARQL queries. The selected domain for the experiment was conference for this domain is generally understandable by most people with a system engineering background.

The accuracy is measured using mapping design worksheets by the participants and integrated ontology as a computation result of auto-generated SPARQL queries by the tool. The ease of use is measured using a questionnaire. This questionnaire is based on the System Usability Scale (SUS) [41] questions and also has ease of use indication question for the UML based abstract semantic mapping representation. The questionnaire was provided to the participants to evaluate the user satisfaction about the abstract semantic mapping representation using UML and the SDI tool.

B. *Evaluation*

The experiment was conducted by 15 participants who were postgraduate level students and academics in engineering department in the university with system engineering experience. There were total 90 semantic mappings conducted by the participants during the experiments: 30 mappings in direct mapping type, 30 mappings in data range mapping type, and 30 mappings in unit transformation mapping. The accuracy of each mapping was examined by comparing the mapping intention in the mapping design worksheet and the integrated ontology as a computation result of auto-generated SPARQL queries by the tool. If the integrated ontology by the auto-generated SPARQL queries matches the mapping intention in the mapping design worksheet, then the mapping is considered accurate. Table 9 shows the results achieved.

TABLE 9 SEMANTIC MAPPING ACCURACY RESULTS

Participant ID	# of correctly captured mappings in the integrated ontology			# of correctly auto-generated mappings
	# of Direct Mappings	# of Data Range Mappings	# of Unit Transformation Mappings	
P01	2	2	2	6
P02	2	2	2	6
P03	2	2	2	6
P04	2	2	2	6
P05	2	2	2	6
P06	2	2	2	6
P07	0	2	2	6
P08	2	2	2	6
P09	2	2	2	6
P10	2	2	1	6
P11	2	2	2	6
P12	2	2	2	6
P13	1	2	2	6
P14	2	2	2	6
P15	2	2	2	6
Total #	**27**	**30**	**29**	**90**

A total of 86 mappings are evaluated as accurate from out of 90 mappings. 4 mappings could not capture accurately participant's mapping intention. From the feedback of participants, it has been identified that there can be graphical intricacy of UML notations for representing multiple semantic mappings in complex ontology relationships. These not accurate mappings were the result of miss-drawing UML notations for the mappings. From the experiment, the number of accurate mappings was 86 out of 90, and this result suggests that the proposed mapping representation could correctly identify the mappings and draw the mappings. In the case of the auto-generation of the mapping syntax from abstract to concrete, as shown in the table 9, the total number of correctly auto-generated mappings was 90 out of 90 mappings (including the ones that did not meet the proper intention of the participant). The accuracy rate of the auto-generation was 100%. This indicates that the tool can correctly generate the mappings automatically from the UML notations.

TABLE 10 EASE OF USE ANALYSIS RESULTS

Participant ID	System Usability Scale Score	
	Abstract Mapping Syntax	SDI Tool
P01	87.5	87.5
P02	70	77.5
P03	85	92.5
P04	92.5	95
P05	97.5	97.5
P06	85	85
P07	77.5	85
P08	70	70
P09	95	100
P10	92.5	100
P11	72.5	85
P12	80	77.5
P13	62.5	52.5
P14	92.5	90
P15	60	95
Average	**81.3**	**86**

The results of the SUS questionnaire are presented in table 10. This presents the opinion of each participant on how easy to use was the UML representation itself and how easy to use was the implemented SDI tool. In the case of ease of use, the average SUS score for the UML based abstract semantic mapping representation marked 81.3 and the tool marked 86. Both SUS scores were higher than 80.3 that means "A" grade which indicates the users are more likely to recommend the representation [42]. These experiment results suggest that proposed abstract semantic mapping representation using UML maintains usability (in terms of accuracy and ease of use) for system engineers.

V. CONCLUSION

This research has identified the core mapping types to be correctly captured by the abstract mapping syntax and reviewed existing semantic mapping syntaxes that tend to rely on a mapping practitioner to understand the idea of ontological concepts. In the research, it is proposed to have a UML-based abstract semantic mapping representation that tries to abstract away the concepts of ontology and is more intuitive to use in order to represent the mapping intentions of domain experts who have system engineering backgrounds. The abstract mapping syntax and tool has been developed into the MOUSE approach, which is usable for system engineers to accomplish their semantic data integration tasks. From the experiment, it is suggested that system engineers can perform the semantic data integration task including ontology mappings without knowing the ontological knowledge by using the proposed UML-based abstract mapping syntax and a tool that they are familiar with, while maintaining accuracy and usability (in terms of ease of use).

The current MOUSE approach is assessed by two criteria. One criterion is the evaluation of the MOUSE approach within the core mapping types, and the other criterion is the evaluation of the MOUSE approach supporting only query-based semantic integration. Since there can be too many possible mapping relationships for different semantic integration situations, there was a need to first scope the mapping relationships and second, create tangible practical semantic integration solutions. Therefore, this research identified and prioritized the mapping relationships that mostly occurred in the industry use case. Three core mapping types were defined to realize the proposed MOUSE approach. This research tried to generalize the use of these core mapping types - in other words, to prove that the core mapping types are usable not only in the telecommunications domain. The core mapping types were applied in another domain (conference) and were proven usable through the examples and experiments conducted by this research.

The MOUSE approach supports only query-based semantic integration. There are three different semantic data integration approaches: (1) Query-based integration using SPARQL, (2) Rule-based integration using SWRL, and (3) Axiom-based integration using ontology. This research selected query-based integration approach to begin the tool development that implements the MOUSE approach because from the related research, query-based integration using SPARQL has proven to be the most practical approach. This does not mean that this research abandoned the other integration approaches. The SDI tool generates an intermediate RIF document because RIF has the potential to interoperate with another integration approach. For example, there is an existing RIF specification for RDF and OWL Compatibility. This strategy potentially enables the transformation between RIF and OWL for axiom-based integration in future.

ACKNOWLEDGMENT

This work is partly funded by an IRCSET / Alcatel Lucent Enterprise Partnership Scheme Postgraduate Research Scholarship, and by Science Foundation Ireland as part of the FAME project (Federated, Autonomic Management of End-to-end communications services - http://www.fame.ie/)

REFERENCES

1. World Wide Web Consortium (W3C), "RDF/XML Syntax Specification (Revised)", W3C Recommendation, http://www.w3.org/TR/rdf-syntax-grammar/, 10 Feb. 2004.

2. World Wide Web Consortium (W3C), "OWL Web Ontology Language Overview", W3C Recommendation, http://www.w3.org/TR/owl-features/, 10 Feb. 2004.

3. Thomas R. Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5, pages 199-220, 1993.

4. Dieter Fensel, "Ontologies: a silver bullet for knowledge management and electronic commerce", Springer, Heidelberg (DE), 2nd edition, 2004.

5. D. O'Sullivan, V. Wade and D. Lewis, "Understanding as We Roam", in IEEE Internet Computing, 11, DOI: 10.1109/MIC.2007.50, p26 - 33, 2007.

6. H. Thomas, D. O'Sullivan and R. Brennan, "Evaluation of Ontology Mapping Representations: a Pragmatic Evaluation", In Workshop on Matching and Meaning, Part of the AISB 2009 Convention, April 9th 2009. Edinburgh, Scotland, 2009.

7. P. Bouquet, M. Ehrig and J. Euzenat, "D2.2.1 Specification of a common framework for characterizing alignment", http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-221v1.pdf, 2005.

8. S. Amrouch and S. Mostefai, "Survey on the literature of ontology mapping, alignment and merging", In IEEE International conference on Information Technology and e-Services (ICITeS), pp. 1-5, 2012.

9. Y. Kalfoglou and M. Schorlemmer, "Ontology Mapping: The State of The Art", The Knowledge Engineering Review Journal (KER), DOI: 10.1017/S0269888903000651, 18(1):1-31, Dec. 2003.

10. S. M. Falconer, N. F. Noy and M. A. D. Storey, "Ontology Mapping - a User Survey", In Proceedings of the Workshop on Ontology Matching (OM 2007), ISWC/ASWC 2007, Busan, Korea, Nov. 2007.

11. I. F. Cruz, C. Stroe and M. Palmonari, "Interactive User Feedback in Ontology Matching Using Signature Vectors", In: Proc. of the 28th Int. Conference on Data Engineering, pp. 1321-1324, 2012.

12. S. Falconer and M. A. Storey, "A cognitive support framework for ontology mapping", In Processings of the 6th International Semantic Web Conference (ISWC), pp. 114-127, 2007.

13. Dale Miller, "Abstract Syntax and Logic Programming", In Proceedings of the First and Second Russian Conferences on Logic Programming, pp. 322-337, Springer-Verlag LNAI 592, Irkutsk and St. Petersburg, Russia, 1992.

14. F. Fondement and T. Baar, "Making Metamodels Aware of Concrete Syntax", European Conference on Model Driven Architecture (ECMDA), LNCS 3748, pp. 190-204, 2005.

15. E. Blomqvist and K. Sandkuhl, "Patterns in ontology engineering: Classification of ontology patterns", In Proceedings of International Conference on Enterprise Information Systems (ICEIS), pp. 413-416, 2005.

16. F. Scharffe and D. Fensel, "Correspondence Patterns for Ontology Alignment", In Proceedings of the 16th International Conference on Knowledge Engineering (EKAW 2008), pp. 83-92, 2008.

17. J. Keeney, A. Boran, I. Bedini, C. J. Matheus and P. F. Patel-Schneider, "Approaches to Relating and Integrating Semantic Data from Heterogeneous Sources", In Proceedings of The 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2011), Lyon, France, 22-27 Aug. 2011.

18. S. Bechhofer et al., "OWL Web Ontology Language Reference", W3C Recommendation, http://www.w3.org/TR/owl-ref/, 10 Feb. 2004.

19. I. Horrocks et al., "SWRL: A Semantic Web Rule Language Combining OWL and RuleML", W3C Member Submission, http://www.w3.org/Submission/SWRL/, 21 May. 2004.

20. H. Boley, "The Rule Markup Initiative: Schema Specification of RuleML 1.0", http://ruleml.org/1.0/, Retrieved 09 Feb. 2011.

21. World Wide Web Consortium (W3C), "SPARQL Query Language for RDF", W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/, 15 Jan. 2008.

22. World Wide Web Consortium (W3C), "Resource Description Framework (RDF): Concepts and Abstract Syntax", W3C Recommendation, http://www.w3.org/TR/rdf-concepts/, 10 February 2004.

23. J. Euzenat, "A format for ontology alignment", http://alignapi.gforge.inria.fr/format.html, Retrieved 09 Mar. 2010.

24. Ontology Alignment Evaluation Initiative (OAEI), "Ontology Alignment Evaluation Initiative Campaign", http://oaei.ontologymatching.org/, 19 June 2012.

25. J. Euzenat, "EDOAL: Expressive and Declarative Ontology Alignment Language", http://alignapi.gforge.inria.fr/edoal.html, Retrieved 13 May 2011.

26. P. Bouquet, F. Giunchiglia, F. V. Harmelen, L. Serafini and H. Stuckenschmidt, "C-OWL: Contextualizing Ontologies", The Semantic Web - ISWC 2003, volume 2870 of Lecture Notes in Computer Science (LNCS), pp. 164-179, FL, USA, Oct. 2003.

27. JENA Apache incubator project, "SPARQL Syntax Expressions", http://openjena.org/wiki/SSE, Aug. 2011.

28. M. J. O'Connor and A. Das, "SQWRL: a Query Language for OWL" OWL: Experiences and Directions (OWLED), 6th International Workshop, Chantilly, VA, 2009.

29. World Wide Web Consortium (W3C), "The Rule Interchange Format", RIF Working Group, http://www.w3.org/2005/rules/wiki/RIF_Working_Group, Oct. 2010.

30. Object Management Group (OMG), "Documents associated with Ontology Definition Metamodel (ODM) Version 1.0", http://www.omg.org/spec/ODM/1.0/, Released May 2009.

31. I. Horrocks and P. F. Patel-Schneider, "Knowledge Representation and Reasoning on the Semantic Web: OWL." Handbook of Semantic Web Technologies, J. Domingue, D. Fensel and J. A. Hendler (Eds.), pp. 365-398, 2011.

32. Object Management Group (OMG), "Object Constraint Language (OCL) - version 2.0", http://www.omg.org/spec/OCL/, Released May 2006.

33. UML® Resource Page, "Unified Modeling Language", http://www.uml.org, Retrieved Jul. 2011.

34. P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar and J. Smith, "UML for ontology development", The Knowledge Engineering Review, 17 (1), pp. 61-64, 2002.

35. D. Gasevic, D. Djuric, V. Devedzic and V. Damjanovi, "Converting UML to OWL ontologies", In Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters, New York, NY, USA, 19-21 May 2004.

36. A. Boran, I. Bedini, C. Matheus, P. F. Patel-Schneider and J. Keeney, "A smart campus prototype for demonstrating the semantic integration of heterogeneous data", In Web Reasoning and Rule Systems, pp. 238-243, Springer Berlin Heidelberg, 2011.

37. A. Boran, C. Matheus, I. Bedini and P. F. Patel-Schneider, "Empirical Analysis of Semantic Techniques applied to a Classification Problem involving Network Performance Data", The 10th International Semantic Web Conference (ISWC), Bonn, Germany, 23-27 Oct. 2011.

38. Alcatel-Lucent Femtocell Test bed, "http://www.alcatel-lucent.com", Alcatel-Lucent 9360 Small Cell.

39. World Wide Web Consortium (W3C), "RIF Core Dialect", W3C Recommendation, http://www.w3.org/TR/rif-core/, 22 Jun. 2010.

40. World Wide Web Consortium (W3C), "RIF RDF and OWL Compatibility", W3C Recommendation, http://www.w3.org/TR/rif-rdf-owl/, 22 Jun. 2010.

41. J. Brooke, "SUS: A quick and dirty usability scale", In B. T. P.W. Jordon, B. A. Weerdmeester and I. L. McClelland (Eds.), Usability evaluation in industry, pp. 189-194, London, England, 1996.

42. J. Sauro, "Measuring usability with the System Usability Scale (SUS)", Retrieved from http://www.measuringusability.com/sus.php, 11 July 2011.