ReShare: An Operational Ontology Framework for Research Modeling, Combining and Sharing

Mohammad Al Boni
Student Member, IEEE
Center for Advanced Vehicular Systems
Mississippi State University, USA
Email: mma201@msstate.edu

Osama AbuOmar
Student Member, IEEE
Center for Advanced Vehicular Systems
Mississippi State University, USA
Email: abuomar@cavs.msstate.edu

Roger L. King
Senior Member, IEEE
Center for Advanced Vehicular Systems
Mississippi State University, USA
Email: rking@cavs.msstate.edu

***Abstract*—Scientists always face difficulties dealing with disjointed information. There is a need for a standardized and robust way to represent and exchange knowledge. Ontology has been widely used for this purpose. However, since research involves semantics and operations, we need to conceptualize both of them. In this article, we propose ReShare to provide a solution for this problem. Maximizing utilization while preserving the semantics is one of the main challenges when the heterogeneous knowledge is combined. Therefore, operational annotations were designed to allow generic object modeling, binding and representation. Furthermore, a test bed is developed and preliminary results are presented to show the usefulness and robustness of our approach.**

***Keywords*—Science formalization, information and knowledge structures, operational ontology, generic modeling**

I. Introduction

Science formalization has become one of the most important challenges. An intensive research in all fields is being conducted continuously. As a result, new algorithms and methodologies are discovered while many old ones have become archived because of their inability to adapt to the new concepts. Also, many researchers are tackling the same problems independently, and the only way to explore their work is through literature. There is no way to know their work while they are conducting it. Therefore, there is an exigent need for a standardized way to describe and share research. Ontology provides a good solution for this problem. Ontology is a knowledge representation mechanism that allows computer-based agents to understand and handle a shared domain-based conceptualization [8]. Ontologies are easy to share and are used in different knowledge engineering fields. For example, it is used in semantic web and biology (e.g., the gene ontology [3]). Ontology has been used as a descriptive language, i.e., we use ontology to describe the semantics (structures, meanings and relations) of a certain problem domain. However, such ontologies have created another problem when dealing with heterogeneous knowledge and some may overlap or differ in the structure of the semantics. This disagreement between ontologies occurs because of the subjectivity of using ontologies, i.e., it is dependent on its authors and how they are using it. Consequently, a need for standardized ontologies emerges. Pease et al.[15] proposed an upper ontology from which others can inherit and extend their own concepts. Even though SUMO helped deal with heterogeneous knowledge from different sources, very specific details still differ. Moreover, such upper ontologies increase the complexity of the structure.

Research experiments not only contain semantics which can be effectively defined and described using regular ontology, but also it contains operations (datasets, source code, analysis, experiments, and/or results). Therefore, an operational ontology needs to be employed instead of the regular descriptive one. Furthermore, to insure robustness and scalability, we need to design our ontology in such a generic way so that it can be easily extended without the need for any change in the core concepts. As a result, we considered the use of a software object oriented (OO) model to insure a high level of abstraction that can suit any research domain. To tackle all these challenges, we proposed ReShare, an operational ontology framework for research modeling, combining and sharing that has the ability to design an operational model for any research domain. The main contribution of this work includes proposing ReShare and developing a test bed to experiment the applicability of ReShare. Using this test bed, we modeled a research study case in the field of materials science, conducted by AbuOmar et al. [1]. In that research, the authors analyzed the vapor-grown carbon nanofiber (VGCNF)/vinyl ester (VE) nanocomposites dataset in order to discover some hidden trends, patterns, and properties associated with this material system. A summary of the statistical experimental design and testing procedures to generate the VGFCNF/VE dataset is given later in this paper. A more detailed discussion can be found in [12–14].

This paper is organized as the following: Section II presents the related works in this field. Section III includes a comprehensive description of ReShare. Experiments, preliminary results and evaluation are presented in section IV. Section V concludes the paper and presents a vision for the future work.

II. Related Work

A number of works have been put forth in the literature regarding the OO modeling using ontologies. Most of these studies focus on static objects modeling and descriptive representation using ontologies. One model was introduced by Siricharoen (2006) where an object has an identity, state and functions. It could be inherited from another object or associated with one or more objects. Values are stored in objects using attributes and properties. These attributes can be primitive types (string, integer ... etc), references to other objects or a set of values of these types [19]. This model was extended and detailed by Siricharoen (2009) to include a name, a set of attributes and operations. Each attribute has a name,

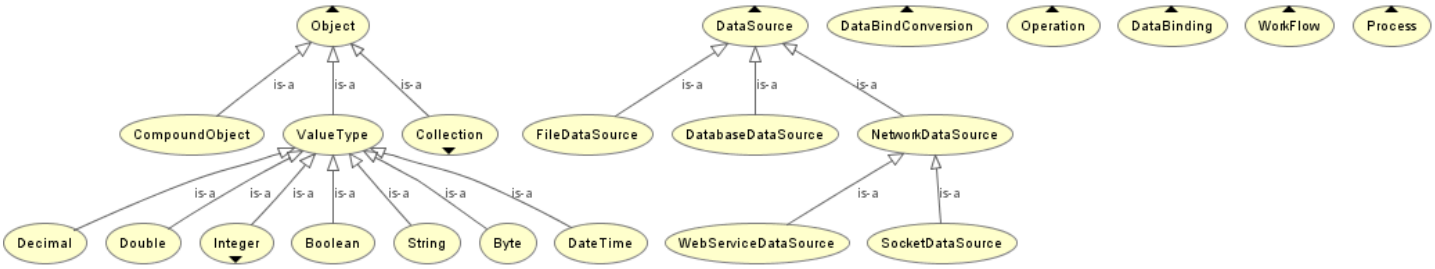

Fig. 1. The main annunciations in the ReShare ontology framework.

type and visibility while operations may contain comments, conditions and/or initial values. Each object may have one or more relationships with other objects such as supertype relation [18]. Another model was described by Batanov and Vongdoiwang (2007): in which an object model consisting of four kinds of entities: object/class, attributes/properties, methods/operations/functions and relations/associations was proposed. In addition, they developed an algorithm that converts a text-based description model of an object to an XML object model through several intermediate steps [4]. Other studies are concerned with the mapping between ontology and the OO model. Evermann and Wand (2005) proposed a set of rules that set the foundation of mapping between objects and an ontology including attributes, associations, composition and aggregation [6]. These works present effective mechanisms for modeling and mapping objects. However, the resultant ontologies, from all of these studies, contain annotations that describe the structure and the associations of objects. None of them define fixed annotations that model any kind of knowledge. Therefore, different research domains will have different object models and different descriptive ontologies. To the best of our knowledge, a high level ontology that can model generic objects has not been reported in the literature.

Ontology was also used in many data source related applications, especially with online data sources and databases. One of the applications was Knowledge Bus in which an application-focused databases were generated from big ontologies such as Cyc and XSB [16]. Also, an API interface was generated to give the users the ability to manipulate the generated databases. Another study was presented by Gali, et al [7]. The authors developed an algorithm for converting descriptive ontology into a relational database. This algorithm created tables which correspond to ontology concepts. Then, relations had been established to specify concepts' associations. Furthermore, similar work was established by Sebestyenova [17] where ontologies were converted not only to a relational database, but also to the OO one. The established databases were used in decision support systems. These models are very suitable for dealing with data sources. However, the outcome of these mechanisms is always descriptive. Because of this, handling and combining data sources from different research domains has become a very difficult task. Moreover, none of these works define a robust model that can be extended to suit new data sources that may be introduced in the future. In this paper, we are addressing these problems by defining a standardized high level operational ontology. This ontology can be used to model and combine knowledge form any research domain. Also, this ontology can be extended

Object Property	Domain	Range
topObjectProperty		
CompoundObjectOperation	CompoundObject	Operation
ObjectBinding	Object	DataBinding
WorkFlowStart	WorkFlow	Process
CollectionItemType	Collection	Object
DataConversionOperation	DataBindConversion	Operation
DataBindConversionDestination	DataBindConversion	DataBinding
OperationParameterCollection	Operation	Collection
CompoundObjectItem	CompoundObject	Object
CollectionItem	Collection	Object
ProcessNext	Process	Process
ProcessOperation	Process	Operation
ProcessPrerequisite	Process	Process
DataBindingConversion	DataBinding	DataBindConversion
OperationReturns	Operation	Object
DataBindingDataSource	DataBinding	DataSource
DataSourceOperation	DataSource	Operation
CollectionOperation	Collection	Operation
ObjectOperation	Object	Operation

Fig. 2. Object property matrix

to include new technologies, such as new database engine, without the need for any change in the core concepts.

III. ReShare Ontology Framework

In this section, we present the annotations of ReShare framework (Figs. 1, 2 and 3). The annotations are mainly divided into five correlated categories: Generic object, data source, data binding, operation and workflow.

A. Generic Object Model

In order to model a generic object for all objects used in any problem, we employed the software OO ideology in our framework. This ideology is generic, robust, dynamic, simply designed and easily understood. The annotation "object" works as the parent for all other instances derived from it. Objects can be of different types (value, collection or compound). First, value type is a variable that can hold a primitive value such as string, integer, boolean ... etc. Another object type is a collection that describes a set of elements. In the collection, we can identify the "ItemType" which is an instance of the Object annotation, e.g., we can model a collection of values, a collection of compound objects or a collection of collections. Finally, compound object represents a domain-based object that contains several items, each can be anyone of the types mentioned above. The main motivation behind using the software OO ideology, herein, is its robustness. Any change to the object model will not require much adaptation. For example, if we have a collection of compound objects and we add a new field to the compound object, then the only

Data Property	Domain	Range
topDataProperty		
ValueTypeSubType	ValueType	string
ProcessAuthor	Process	string
Value	ValueType	
DoubleValue	Double	double
StringValue	String	string
DateTimeValue	DateTime	dateTime
ByteValue	Byte	byte
BooleanValue	Boolean	boolean
DecimalValue	Decimal	decimal
IntergerValue	Integer	integer
OperationInnerCode	Operation	string
DataBindingID	DataBinding	integer
WebSourceDestinationEndPoint	NetworkDataSource	string
OperationSubType	Operation	string
CollectionSubType	Collection	string
OperationSourceType	Operation	string
DatabaseConnectionString	DatabaseDataSource	string
DataBindConversionID	DataBindConversion	integer
WorkFlowName	WorkFlow	string
ObjectID	Object	integer
DataBindingName	DataBinding	string
WorkFlowID	WorkFlow	integer
DataBindingInsertOperation	DataBinding	string
DataSourceName	DataSource	string
FileDataSourcePath	FileDataSource	string
DataBindingBindingOperation	DataBinding	string
WebServiceEncoding	WebServiceDataSource	string
OperationID	Operation	integer
ValueTypeBindingField	ValueType	string
FileDataSourceFormat	FileDataSource	string
WebServiceServiceName	WebServiceDataSource	string
DataSourceSubType	DataSource	string
DataBindConversionName	DataBindConversion	string
ObjectAccessability	Object	string
DatabaseEngineType	DatabaseDataSource	string
CompoundObjectID	CompoundObject	integer
OperationAccessability	Operation	string
DataBindingUpdateOperation	DataBinding	string
ProcessName	Process	string
DataBindingDeleteOperation	DataBinding	string
DataSourceID	DataSource	integer
ProcessID	Process	integer
ObjectSubType	Object	string
WebServiceURL	WebServiceDataSource	string
CollectionID	Collection	integer
DataBindConversionRate	DataBindConversion	string
ObjectName	Object	string
ValueTypeID	ValueType	integer
ProcessPresentOutput	Process	boolean
DataBindingBindingItems	DataBinding	string
DataBindingUnit	DataBinding	string
WorkFlowVersion	WorkFlow	string
OperationName	Operation	string

Fig. 3. Data property matrix

required adaptation is adding new annotations for the new field. Afterwards, the change will take effect in place, and the collection will contain the compound object's new design without any change to its annotations or associations.

B. Data Source Model

The proposed data source model is extendable, easy to use and gives the ability for on-the-fly dynamic data binding. All data sources will be handled through one root node "DataSource". Data sources can be of different types such as "FileDataSource", "NetworkDataSource", "DatabaseDataSource" or it can be any data source nodes added for future use. Moreover, using the operation scheme (discussed later in this section), we can define all the operations that are required to give an interface for that data source. Operations can be of any type: Read, Write, Update or Delete. Multiple operations of each type can also be defined. This model can easily be extended. Whenever a new data source is added, we just need to plug it in a suitable location in the data sources tree and define the operations that allow the system to extract and manipulate data to/from it.

C. Data Binding Model

In this section we will discuss the binding between the object model and the data sources. The data binding model consists of two main concepts "DataBinding" and "DatabindConversion". Data binding relates objects to a certain data source and defines the operations which need to be called from the data source in order to extract and manipulate the corresponding data. Also, each data binding can define a specific unit. Using the unit and the data bind conversion annotations, we can combine and compare data from different data sources. For example, we can compare the temperature of two materials, each one is stored in a different data source and each has its own different unit. Moreover, since we can associate data bindings with specific object models or items of the object models, we can partially bind objects from different data sources and combine the results to create fully objects.

D. Operation Model

The proposed operation model represents a generic function that takes a collection of parameters and returns an object (value, collection or compound) as an output. Operations can be of different categories such as read, insert, update, delete, operational, analytical ...etc. The operation source code can be written in any coding language (C++, C#, Java ...etc). This parameters' collection can be defined using the same object models discussed previously.

E. Workflow Model

A study may include a number of workflows, each represents a certain experiment. Each workflow may consist of many steps and these steps can be sequential or concurrent. In order to satisfy these requirements, two annotations were used ("WorkFlow" and "Process"). A workflow will start from certain processes, and a dependency diagram is determined by specifying each process's prerequisites and its next processes. Each process executes an operation designed according to the operation model and returns an object (value, collection or compound). This object can be fed as an input for the next process. Also, processes can visualize their results using external visualization engines.

In the following section, a research case study, modeled using ReShare, in materials science is presented. This example will help to understand the correlation and associations between all of the previous categories.

IV. Experimental Results and Evaluation

A research experiment conducted by AbuOmar et al. [1] is modeled using the proposed ReShare annotations. In this section, the experiment is briefly described in terms of statisical design, the corresponding ReShare model is built and the outcome of this experiment is shown.

A. Statistical Experimental Design

The effect of five input design factors on the viscoelastic properties (storage, loss modulus and tan delta) of VGCNF/VE nanocomposites were investigated using a general mixed-level full factorial experimental design [11]. These carefully selected factors, based on the state-of-the-art formulation and processing procedures, included:

(1) VGCNF type (designated as A),
(2) use of a dispersing agent (B),
(3) mixing method (C),
(4) VGCNF weight fraction in parts per hundred parts of resin (phr) (D), and
(5) the temperature (E) used in dynamic mechanical analysis (DMA) testing.

Experimental design factors and their associated levels are given in Table I.

TABLE I. THE EXPERIMENTAL DESIGN FACTORS AND THEIR LEVELS.

Factor designation	Factors	Level				
		1	2	3	4	5
A	VGCNF type	Pristine	Oxidized	-	-	-
B	Use of dispersing agent	Yes	No	-	-	-
C	Mixing method	US[a]	HS[b]	HS/US	-	-
D	VGCNF weight fraction (phr[c])	0.00	0.25	0.50	0.75	1.00
E	Temperature (^{o}C)	30^{o}C	60^{o}C	90^{o}C	120^{o}C	-

[a] Ultrasonication
[b] High-shear mixing
[c] Parts per hundred parts of resin

A total of $2\times2\times3\times5\times4 = 240$ "treatment combinations" (different combinations of the factor levels in Table I) were randomized to eliminate bias in preparing the specimens. Each treatment combination resulted in three specimens prepared from the same material batch [13, 14]. Each specimen was tested using a dynamic mechanical analyzer (single cantilever/flexure mode) to measure average storage modulus, loss modulus, and tan delta for each treatment combination. Storage and loss modulus are dynamic mechanical properties and indicative of the polymer nanocomposite's stiffness and energy dissipation capability, respectively [1].

B. ReShare Model

Herein, the overall experiment model consists of two concurrent workflows: principal component analysis (PCA)[9] followed by fuzzy C-means (FCM) clustering[5], and a self-organizing map (SOM)[10] study. Both workflows use the same dataset, and they are applied on the same object model. Therefore, we will combine them into one workflow with two separate process sequences.

1) Object Model: the objects in this experiment belong to two categories:

(a) Main Object: the model, used for this experiment, consists of 8 dimensions and both PCA and SOM take a set of records of that model. Therefore, the main object in our model (VGCNF/VE) is a collection of compound objects (Fig. 4). VGCNF/VE has eight items and it is bound to a data source through the data binding model described below. Detailed description of the VGCNF/VE object is presented in Table II.

(b) Auxiliary Objects: few secondary objects were modeled to assist the execution of the model. First,

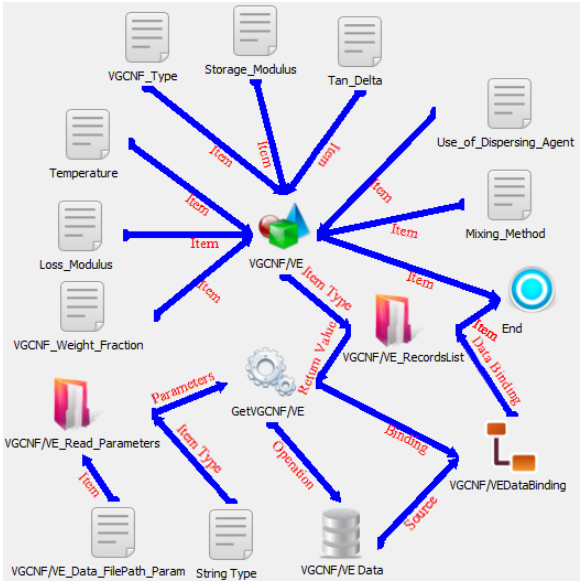

Fig. 4. VGCNF/VE object and its data source and data binding model.

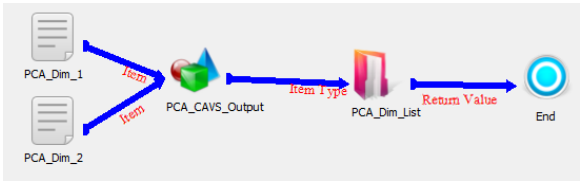

Fig. 5. PCA parameters object model

VGCNF/VE_Read_Parameters is a collection used as an input for the GetVGCNF/VE operation (Fig. 4). It contains one ValueType item that stores the dataset physical location. PCA_Dim_List (Fig. 5) is another collection which is used to store the output of the PCA operation. PCA is an analytical study that converts N-dimensional data into M dimensions (where $M < N$). However, for easier visualization purposes, PCA was used to represent the data in 2 dimensional space. PCA_Dim_List has two items, each of them stores the value of one dimension. In Figure 6, we have three more auxiliary objects. PCA_CAVS_Parameters encapsulates the VGCNF/VE collection and provides it as an input to the PCA_CAVS operation. Similarily, SOM_CAVS_Parameters encapsulates the collection and provides it as an input to the SOM_CAVS operation.Finally, FCM_CAVS_Parameters provides the PCA_Dim_List, resulted from the PCA, and the number of clusters to the FCM_CAVS operation.

TABLE II. VGCNF/VE OBJECT

Item	Object attributes		Valuetype attributes
	SubType	accessibility	SubType
VGCNF_Type	ValueType	public	integer
Use_of_Dispersing_Agent	ValueType	public	integer
Mixing_Method	ValueType	public	integer
VGCNF_Weight_Fraction	ValueType	public	double
Temperature	ValueType	public	double
Storage_Modulus	ValueType	public	double
Loss_Modulus	ValueType	public	double
Tan_Delta	ValueType	public	double

2) Data Source and Data Binding Model: the dataset, used in this experiment, is stored as a text file. We defined this data

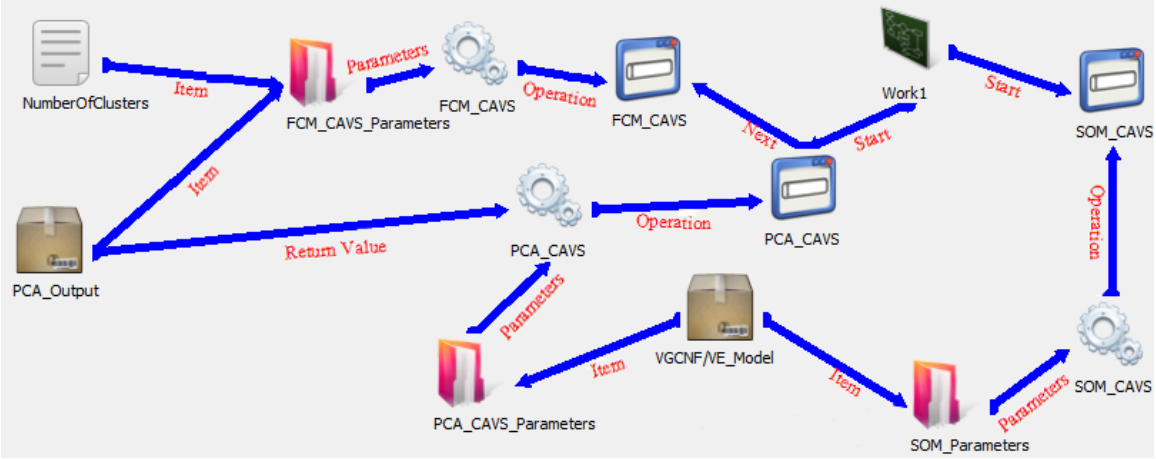

Fig. 6. Experiment workflow and dependency between its processes.

source as an instance of the "FileDataSource" and associated it with a data binding and a read operation's annotations. The data binding ontology describes how the GetVGCNF/VE operation will be executed on the data source in order to extract a collection of VGCNF/VE's compound object (Fig.4).

3) Workflow and Operation Model: we have mainly one workflow that starts with two separate process sequences: PCA followed by FCM and SOM. Also, we have a set of operations for reading the dataset and calling the experiment's functions. PCA, FCM and SOM were originally written in Matlab. Without any change to the original source code, we used a C# code (PCA_CAVS, FCM_CAVS and SOM_CAVS respectively) that calls the Matlab engine, passes the corresponding parameters, executes the functions and returns their outcome to our model. The final results are shown in Figs. 7 and 8. Note, the images, in Figs. 7 and 8, were generated by the Matlab R2012a engine, then exported and displayed in our test bed on-the-fly. These results matched with the original experiment results in [1]. In addition, Figures 4 through 8 are taken from our test bed.

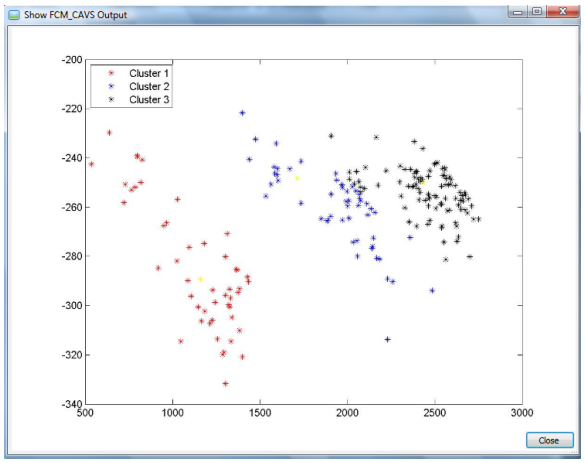

Fig. 7. Visualized FCM results.

C. Evaluation

The most important issue which needs to be addressed in the evaluation is the robustness of ReShare. How will the

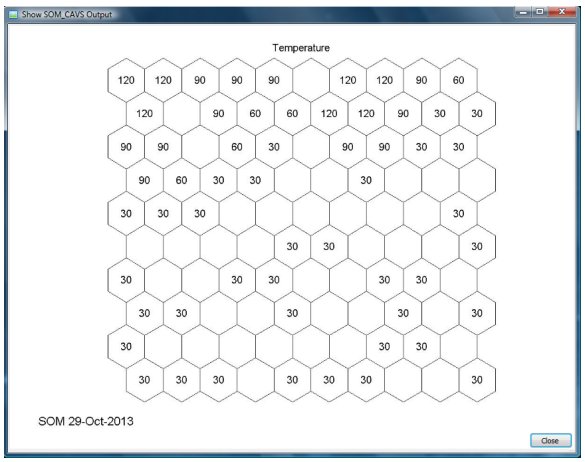

Fig. 8. Visualized SOM results.

system react to any future adjustment in the objects' model? Any change in the object model (adding, editing and deleting dimensions) will not affect the system functional behavior. In the previous related work [4, 6, 18, 19], any change in the object model will require building the ontology from scratch. Also, in the previous studies, the authors generate a descriptive ontology. Therefore, it will be a difficult task to combine models from different research experiments. Using the proposed model in this paper, any change in model(s) that describe the experiment(s) will only require few modifications, and no change is required in the core ontology. Likewise, combining models from different research experiments has become easy to perform since we only need to combine dimensions and adjust the data binding model. Also, what if a new technology is introduced? how will the system adapt to the new technologies? The related work in the area of mapping between data sources and ontologies [7, 16, 17] proposed specific mechanisms for certain data sources such as a database. If a new data source is introduced in the future, these mechanisims will no longer become applicable. Therefore, researchers need to develop new ways to deal with these new types of potential data sources. Moreover, none of these studies dealt with heterogeneous data sources. Thus, if

no robust framework is defined, it will be difficult to combine the research studies on the existing technologies with those on the new technologies. ReShare provides a solution for this problem through the operation model. For example, if a new type of data source is introduced, only new annotations will be plugged into the data source ontology tree. Then, suitable operations should be defined to deal with the new data source, and objects can be bound to the new data source on-the-fly without changing the core of our ontology structure.

V. Conclusion and Future Work

In this paper, we proposed ReShare: an operational ontology framework that provides scientists with the ability to build, combine and share their research experiments. In this framework, we employed the software OO methodologies to create a highly robust and adaptive ontology. This ontology can be used to model experiments in any research domain. We also developed a test bed, and we have successfully used it to model a research experiment in the field of materials science. This task was performed only by using few high level annotations and the results were also displayed in our test bed. Moreover, ReShare models can be exchanged between researchers by sharing the ontology which will facilitate the process of understanding and extending models that were previously designed. Furthermore, the main advantage of using ontology over the traditional XML is that it provides the ability to perform semantic search. This feature will increase the practicality of our work, and researchers can effectively look for models built using ReShare. However, few things need to be addressed for future work. First, in the current framework, the results are externally generated and then exported to our test bed. We need to propose a set of annotations that will help to formalize the visualization processes. These annotations will also help researchers to perform semantic search on the results that are generated by ReShare. Moreover, since we have a huge amount of descriptive ontologies, we need to design an algorithm that utilizes these existing ontologies to define operational models.This algorithm will also need to handle different ontologies that describe the same area of research. To do this task, we are planning to adopt few ontology matching and alignment techniques, such as [2], to aggregate overlapped ontologies in order to build a consistent and complete object model. Furthermore, we would like to develop an algorithm that converts the operation model, designed by ReShare, to a descriptive one. This will be very useful to generate ontologies in fields were such descriptive ontologies are not available.

Acknowledgment

The first author would like to acknowledge and thank the Fulbright scholarship program for its financial support of the work performed herein.

References

[1] O. AbuOmar, S. Nouranian, R. King, J. Bouvard, H. Toghiani, T. Lacy, and C. P. Jr. Data mining and knowledge discovery in materials science and engineering: A polymer nanocomposites case study. *Advanced Engineering Informatics*, 2013.

[2] M. Al Boni, D. T. Anderson, and R. L. King. Constraints preserving genetic algorithm for learning fuzzy measures with an application to ontology matching. In *2013 Third Annual World Conference on Soft Computing (WCSC 2013)*, San Antonio, USA, Dec. 2013.

[3] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1), 2000.

[4] D. N. Batanov and W. Vongdoiwang. Using ontologies to create object model for object-oriented software engineering. In *Ontologies*, pages 461–487. Springer, 2007.

[5] J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[6] J. Evermann and Y. Wand. Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering*, 10(2):146–160, 2005.

[7] A. Gali, C. X. Chen, K. T. Claypool, and R. Uceda-Sosa. From ontology to relational databases. In *Conceptual Modeling for Advanced Application Domains*, pages 278–289. Springer, 2004.

[8] T. R. Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 1993.

[9] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, Oct. 2002.

[10] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[11] D. C. Montgomery. *Design and analysis of experiments*. John Wiley & Sons, Hoboken, NJ, 7th ed. edition, 2009.

[12] S. Nouranian. *Vapor-grown carbon nanofiber/vinyl ester nanocomposites: Designed experimental study of mechanical properties and molecular dynamics simulations*. PhD thesis, Mississippi State University, 2011.

[13] S. Nouranian, T. E. Lacy, H. Toghiani, C. U. Pittman, and J. L DuBien. Response surface predictions of the viscoelastic properties of vapor-grown carbon nanofiber/vinyl ester nanocomposites. *Journal of Applied Polymer Science*, 2013.

[14] S. Nouranian, H. Toghiani, T. E. Lacy, C. U. Pittman, and J. Dubien. Dynamic mechanical analysis and optimization of vapor-grown carbon nanofiber/vinyl ester nanocomposites using design of experiments. *Journal of Composite Materials*, 45(16):1647–1657, 2011.

[15] A. Pease, I. Niles, and J. Li. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *Working notes of the AAAI-2002 workshop on ontologies and the semantic web*, volume 28, 2002.

[16] B. Peterson, W. A. Andersen, and J. Engel. Knowledge bus: Generating application-focused databases from large ontologies. In *Proceedings of the 5th Workshop KRDB-98, Seattle, WA, USA*. Citeseer, 1998.

[17] J. Sebestyénová. Domain ontology based object-oriented and relational databases. In *Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, pages 324–329. World Scientific and Engineering Academy and Society (WSEAS), 2005.

[18] D. W. V. Siricharoen. Ontology modeling and object modeling in software engineering. *International Journal of Software Engineering and Its Applications*, 2009.

[19] W. V. Siricharoen. Ontologies and object models in object oriented software engineering. *IAENG International Journal of Computer Science*, 33(1):19–24, 2007.