Available online at www.sciencedirect.com
ScienceDirect
Computers and Chemical Engineering 30 (2006) 1482–1496
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Ontological informatics infrastructure for pharmaceutical product development and manufacturing

Venkat Venkatasubramanian *, Chunhua Zhao [1], Girish Joglekar, Ankur Jain, Leaelaf Hailemariam, Pradeep Suresh, Pavankumar Akkisetty, Ken Morris, G.V. Reklaitis

Laboratory for Intelligent Process Systems, School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA

Received 7 March 2006; received in revised form 23 May 2006; accepted 24 May 2006
Available online 14 July 2006

Abstract

Informatics infrastructure plays a crucial role in supporting different decision making activities related to pharmaceutical product development, pilot plant and commercial scale manufacturing by streamlining information gathering, data integration, model development and managing all these for easy and timely access and reuse. The foundation of such an infrastructure is the explicitly and formally modeled information. This foundation enables knowledge in different forms, and best manufacturing practices, to be modeled and captured into tools to support the product lifecycle management. This paper discusses the development of ontologies, Semantic Web infrastructure and Web related technologies that make such an infrastructure development possible. While many of the issues addressed in this paper are applicable to a wide spectrum of molecular-based products, we focus our work on the development of pharmaceutical informatics to support Active Pharmaceutical Ingredient (API) as well as drug product development as case studies to illustrate the various aspects of this infrastructure.

Keywords: Pharmaceutical informatics; Product design; Process development; Ontology; Semantic Web

1. Introduction

The evolution of commercial scale product and process development typically goes through the following stages after the viability of a newly discovered molecule is established: laboratory scale, pilot plant scale and commercial scale manufacturing. Laboratory scale experiments are used to determine various synthesis routes and characterize key steps in each route. Pilot plant studies provide detailed understanding of processing steps of the selected route and provide data needed for scale-up, culminating in a commercial scale process. A commercial scale process provides valuable manufacturing related information useful for debottlenecking and productivity improvement. These three processing stages, and the major tasks contained therein, are closely related to each other through various information feedback loops as shown in Fig. 1.

As Fig. 1 suggests, development of a process for an active pharmaceutical ingredient (API) and its dosage form(s) evolves in stages, requiring participation from several groups and uses a wide range of application tools. Recent progress in the use of new process analytical technologies (PAT) (FDA, 2005) has led to a better monitoring and gathering of data of the physical and chemical phenomena. Techniques such as mass spectrometry for gathering reaction kinetics data, Lasentec Focused Beam Reflectance Measurement (FBRM) instrument (Sistare, Berry, & Mojica, 2005) for crystal growth, and so on have made this possible. For laboratory information integration, solutions like LIMS (Paszko & Pugsley, 2000) and e-Lab Notebook (Zall, 2001) have been developed. However, a systematic way to convert this raw data gathered from PAT to information and first principles knowledge that can be used for real-time decision making is lacking.

During process development, a staggering amount of information of different types, ranging from raw data to lab reports to sophisticated math models, is shared and revised by computational tools in each stage. Subsequent to process development, technical specifications and reports must be devel-

* Corresponding author. Tel.: +1 765 494 0734; fax: +1 765 494 0805.
E-mail address: venkat@ecn.purdue.edu (V. Venkatasubramanian).
[1] Present address: Bayer Technology and Engineering (Shanghai) Co. Ltd., Shanghai 201507, China.

0098-1354/$ – see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2006.05.036

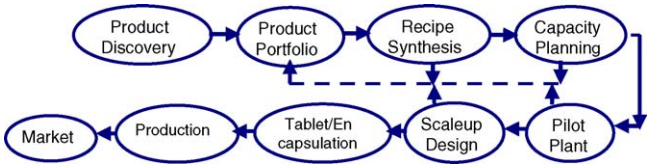

Fig. 1. Information flows in pharmaceutical product and process development.

oped to satisfy regulatory requirements. However, due to the incompatibility among such diverse types of data, information and knowledge, islands of automation exist, but a comprehensive, integrated decision support environment does not exist. With the onset of information and knowledge explosion, it is clear that we need intelligent software systems to effectively manage and access information for efficient decision making.

In the chemical process development domain, several attempts have been made towards process information integration, such as *KBDS* (Bañares-Alcántara & Lababidi, 1995), *MDOOM* (Burkett & Yang, 1995; McGreavy, Wang, Lu, & Naka, 1995) and *PROART/CE* (Jarke & Marquardt, 1996). Schneider and Marquardt (2002) presented a conceptual lifecycle integration model based on the mapping of workflow models into information models describing chemical processes and entities. These are important first attempts recognizing the need for an informatics framework in modeling complex operations. However, these solutions are limited in their description of the relations between the different information entities and in their application to the pharmaceutical domain. Additional limitations include lack of formalization for knowledge modeling and the relation to data. While the general concepts of information modeling are common across these prior attempts and ours, the challenges are in the details posed by the particular application domain. From that perspective, this is the first comprehensive informatics framework in the pharmaceutical product development domain. In this paper, we present an overview of our approach to address these challenges. In our work, we start from modeling the data/information/knowledge as well as their flows in the entire process development. An informatics infrastructure is developed to support decision making spanning the entire process, including drug product formulation design, process simulation and process safety analysis for API process as well as drug product development.

The rest of this paper is organized as follows: a brief review of related work in information modeling is presented and the proposed approach is discussed. The foundation of this approach is the formal modeling of the domain information. The concept of ontology is introduced; issues of ontological language are discussed. Ontology development is then discussed using ontology for recipe information and material properties as case studies. Uses of the ontology for information management and information sharing are discussed. The implication of service-oriented application and its impact on tools development is discussed. Based on the formal modeling of information, various forms of knowledge, such as guidelines and mathematical equations, are captured and modeled to support the decision making.

2. Information modeling

Over the past 20 years, several tools have been developed to support chemical product/process development and manufacturing. As an example, we examine three tools which are used in pilot plant operation: BatchPlus (AspenTech, 2005) for recipe development and mass/energy balance calculations, Batches (Batch Process Technologies, 2005) for dynamic simulation of batch processes, and PHASuite (Zhao, 2002) for process safety analysis. The information required for these three tools is very similar, namely information on materials, equipment and recipe. Furthermore, exchange of information between these tools is crucial. The mass and energy balance information generated by simulation packages is required for safety analysis packages. Each tool, however, has its own *syntax* (format of the information) and *semantics* (meaning of the information) to describe it, thus making it hard to communicate with each another. Hence, the form in which the information is required and created by these tools is *application centric*. To store information BatchPlus uses a relational database, PHASuite uses a database and object binary serialization, while Batches uses plain text files. Not only do these tools use similar information, they also generate similar information, for example, the mass and energy balances, cycle time statistics and so on.

In the application-centric view, the scope and representation of a process are influenced by the tool being used. As a result, complete representation of a process is often not available. For example, a simulation tool does not need to have material safety related data for its execution, but this information is crucial to process safety analysis. The biggest challenge is to overcome the syntactic and semantic differences of information so that the information can be shared and reused among different tools (TopQuadrant, 2003).

To share information between tools, a three-step process is required (Zhao, Bhushan, & Venkatasubramanian, 2003):

1. Access and extract information, based on the understanding of syntax and semantics of the information source.
2. Translate the acquired information into semantics understood by the information destination.
3. Store the information based on the syntax defined in the destination.

Clearly, this scheme to share information is error prone, requires expertise in the source and destination tools, and is very time consuming. Also, with the different versions of the same information, managing changes to information becomes very challenging. In this environment, developing new applications also becomes difficult since a new syntax for the process information must be created. The lack of a coherent and unified process view results in islands of information. This limits the information sharing as well as function sharing among tools, thus limiting the use of these tools in decision support and learning.

The key to resolving this problem is to identify the various categories of information used for defining a process and used

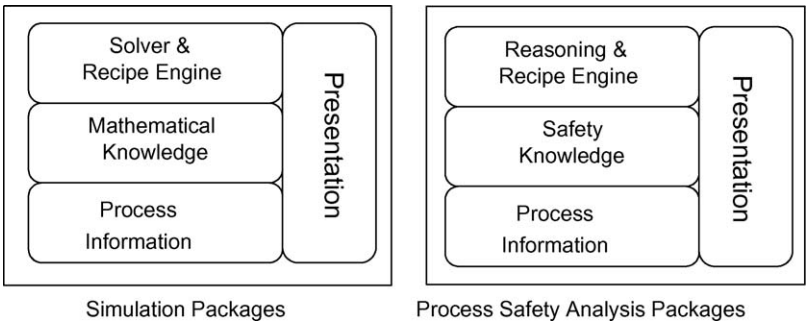

Fig. 2. Similarities of overall structure and functional components in different tools.

in decision support. Subsequently, suitable mechanisms must be developed for managing the information and delivering the right information for the intended use.

As illustrated in Fig. 2, various categories of tools have similar overall structure and functional components. On top of the information layer is typically the knowledge layer, which consists of quantitative models for unit operations in simulation packages, or qualitative cause–effect models in safety analysis packages. The engine layer contains the solver in simulation packages, or reasoning engine for safety analysis package. The presentation layer provides an interface to access and provide information. Failure to recognize the structural similarities has precluded the development of an infrastructure for information sharing, and has resulted in software designs where the knowledge is 'trapped' in each tool, making it very difficult to create, customize, share and manage the knowledge. To address these challenges, an information-centric approach has been proposed for process development. In this new paradigm, the underlying information is modeled *explicitly*, independent of tools that use the information. Instead of encoding information in objects in a particular programming language or tool specific constructs, the information is *explicitly described*. Knowledge is also explicitly modeled.

2.1. Review of related work

Perhaps the first attempt in chemical engineering towards a formal informatics framework for process design was by Bañares-Alcántara and Lababidi (1995) who proposed *KBDS* to integrate the process design space with an object-oriented description of processing units and a design history archive. McGreavy et al. (1995) developed a multi-dimensional object-oriented conceptual model (*MDOOM*) for software tool development consisting of models of physical descriptors like plant topology and process descriptors like pressure drop communicating through a common information exchange standard called the STandard for Exchange of Product model (*STEP*). Jarke and Marquardt (1996) presented *PROART/CE*, which is a tool integration environment based on a definition of representation of data and specification of the physical structure of the units modeled. Schneider and Marquardt (2002) presented a conceptual lifecycle model that covers the workflow with its activities and their order in a (*work*) *process model* and the results produced in a *product* (*data*) *model*. The (*work*) *process model* used the *C3* formalism (Killich et al., 1999), whose major elements are roles, activities, control flow, input and output info, tools and synchronous communication. The *product* (*data*) *model* used the Conceptual Lifecycle Process Model (*CLiP*) (Schneider and Marquardt, 2002). Activities were organized by role and temporal order in *C3* and mapped to *CLiP*, which has been developed to serve as ontology for chemical processes and the associated design processes and characterizes input and output information more precisely. Integration of product data was implemented through Description Logics, which capture the entity-relationships as well as object-oriented data models. For integration of tools, the authors suggested either product data centered integration (where various tools are interfaced with converters using a global schema) or control centered integration (data exchange through standard formats).

As mentioned above, some of the generic informatics concepts we use have been used by these earlier works, but these systems are not directly applicable to the pharmaceutical domain. For instance, the range of material properties required in pharmaceutical domain is much wider, such as solid flow properties, mechanical properties of powders, crystalline properties of solids and so on. Additionally, the need to include access to experimental information is crucial in this domain. Since most of the processing is done in batch mode and according to specific recipes, the recipe information modeling is also a necessary component of the overall infrastructure.

2.2. Ontology for information modeling

To describe the information explicitly, the syntax as well as semantics for the information must be defined. The explicit description of domain concepts and relationships between these concepts is known as *ontology* (Gruber, 1993). A detailed definition given by Studer, Benjamins, and Fensel (1998) which is based on Gruber's (1993) definition is "An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group."

Ontology has some similarities and at the same time, many differences compared to other information modeling methodologies. Compared to databases, both of them are used by applications at run-time for queries and reasoning, however, relationships are defined as first-class constructs only in ontologies. Similar to object models in object-oriented programming languages (such as C++, C# or Java), ontologies describe classes and attributes, on the other hand, ontologies are set-based and dynamic. Various methodologies have been proposed to model ontologies, including frames and first order logic (Gruber, 1993) and Description Logic (DL) (Branchman & Schmolze, 1985). Many different languages have been developed to encode the ontologies, including Ontoligua (Farquhar, Fikes, & Rice, 1997), LOOM (MacGregor, 1991), CycL (Lenat & Guha, 1990) and OKBC (Chaudhri, Farquhar, Fikes, Karp, & Rice, 1998).

Recent developments in the field of ontology have created new software capabilities that facilitate the implementation of the proposed informatics infrastructure. The shared understanding is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. Ontologies can be used to describe the semantics of the information sources and make the contents explicit, thereby enabling integration of existing information repositories, either by standardizing terminology among the different users of the repositories, or by providing the semantic foundations for translators. Compared to a database schema which targets physical data independence, and an XML Schema which targets document structure, an ontology targets agreed upon and explicit semantics of information. As a result, while the functionalities of this infrastructure can be implemented in a traditional client–server framework, the main benefits of this ontology-driven architecture are its openness and semantic richness.

2.3. Ontological languages: XML, RDF and OWL

In order to model the information explicitly and formally, we must first decide the modeling language to use. As discussed above, several ontological languages have been developed in the Artificial Intelligence (AI) community. The World Wide Web Consortium (W3C) has proposed several markup languages based in these languages for Web environment. Three languages are briefly discussed here: XML, RDF and OWL (see URL). As shown in Fig. 3, these languages have different levels of semantic support and expressive powers.

Extensible Markup Language (XML) is a meta-language for markup, which does not have a fixed set of tags but allows users to define tags of their own. It defines the common syntax for various languages. XML is originally designed to describe a document. As an XML Schema defines the structure of XML documents, an XML document can be validated according to the corresponding XML Schema. XML Schemas have been developed for several domains, such as Batch Markup Language (BatchML, see? http://www.wbf.org/displaycommon.cfm?an=1&subarticlenbr=43) for batch process information, Mathematical Markup Language (MathML, see http://www.w3.org/Math/) for mathematics information, Analytical Information Markup Language (AnIML, see http://animl.sourceforge.net/) for analytical information and Chemical Markup Language (CML, see http://www.xml-cml.org/) for molecule information.

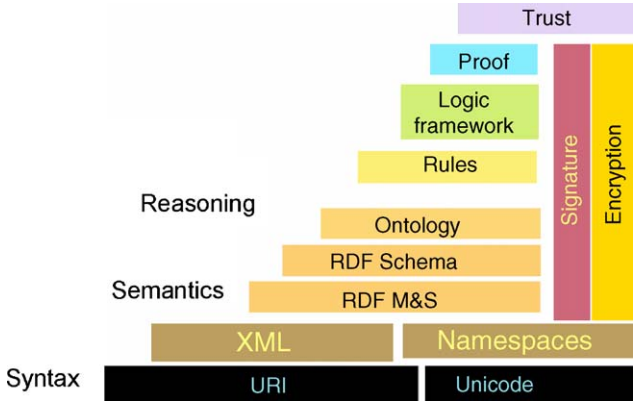

Fig. 3. Stack of Semantic Web Markup Languages.

Even though XML has become a uniform data exchange format between applications, it does not provide any means of defining the *semantics* (meaning) of the information. Furthermore, XML is sufficient to describe a tree structure, but not for general information relations which requires a graph structure. We will use a simple example from API recipe to discuss the capabilities of the three languages. In this example, we define the concepts of *UnitProcedure* and *Operation* and the relations that *UnitProcedure* contains *Operations*. We also need to define the *Charge* as a subclass of *Operation*. The graph representation of these concepts is illustrated in Fig. 4.

Fig. 5 shows the tree representation of this information model. The relation is indeed defined within the *UnitProcedureType* as *UnitOperation* contains *Operations*. *Charge* is defined as an extension of *Operation*.

Resource Description Framework (RDF) has been developed based on XML syntax to define the graph structure of the information. The fundamental concepts of RDF are resources, properties and statements. In RDF, every Web resource (can be thought as an object) has a Universal Resource Identifier (URI). Properties are a special kind of resources to describe relations between resources. Properties are also defined by URIs. Statements assert the properties of resources. A statement is an object–attribute–value triplet, consisting of a resource, a property and a value. Values can either be resources or literals. Similar to XML Schema, RDF Schema (RDFS) defines the structure of RDF. Concepts like UnitProcedure and Operation are defined by URI of http://www.purdue.edu/pharmainfo/recipe#UnitProcedure and http://www.purdue.edu/pharmainfo/recipe#Operation.

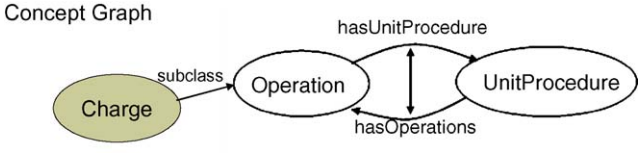

Fig. 4. Concept graph of an example.

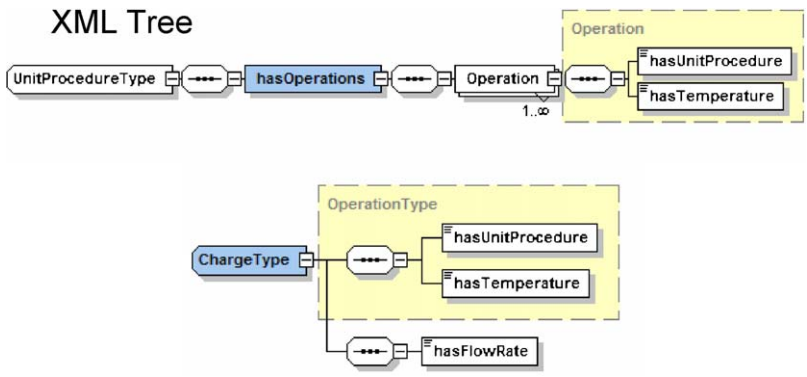

Fig. 5. Define relation of UnitProcedure and Operation in XML.

The relation is defined as property which has also a URI linking the concepts of UnitProcedure and Operation as shown in Fig. 6.

The expressivity of RDF and RDFS is limited by design. RDF/RDFS allows the representation of some ontological knowledge. The main modeling primitives of RDF/RDFS concern the organization of vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classes (Antoniou & Harmelen, 2004).

An ontology language should allow users to write explicit, formal conceptualizations of domain information. Besides a well-defined syntax, a formal semantics, efficient reasoning support, sufficient expressive power are also required. We used Web Ontology Language (OWL) for information modeling. OWL is based on the description logic theory, and is a vocabulary extension of RDF. OWL formalizes a domain by defining classes, properties of these classes and relations between them. OWL can also define individuals and assert properties about them, and furthermore reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language. As shown in Fig. 7, similar to RDFS, the *hasOperation* is defined as a property, and more than that, it is further defined as ObjectProperty which links two concepts. It also defines the Inverse Functional relations with another property: if *hasOperation* describes *UnitProcedure1* (an instance of UnitProcedure) contains *Operation1* (an instance of Operation), the hasUnitProcedure property of *Operation1* uniquely links back to *UnitProcedure1*, as illustrated in Fig. 7. The relations between *Charge* and *Operation* is defined as *subClassOf*. The graph structure generated from the information modeling is also presented in Fig. 7.

```
xml:base = "http://www.purdue.edu/pharmainfo/recipe#">

<rdfs:Class rdf:ID="UnitProcedure">
</rdfs:Class>

<rdfs:Class rdf:ID="Operation">
</rdfs:Class>

<rdf:Property rdf:ID="hasOperations">
   <rdfs:domain rdf:resource="#UnitProcedure"/>
   <rdfs:range rdf:resource="#Operation"/>
</rdf:Property>
```

Fig. 6. Define relation of UnitProcedure and Operation in RDFS.

2.4. Ontology building

In this section, the workflow, fundamental rules and best practices that we have found useful when developing ontologies are discussed using the general recipe ontology as an example. As discussed earlier, ontology models a particular domain using the constructs provided by a suitable ontology language such as OWL. Ontology building is an evolutionary design process consisting of proposing, implementing and refining classes and properties that comprise an ontology (Noy & McGuinness, 2001). In a sense, there is no one "correct" way of developing an ontology. Some of the typical steps are given below.

2.4.1. Step 1: determine the domain and scope of the ontology

The domain of ontology encompasses the information and relations that will be modeled using that ontology. The general recipe ontology described here was developed so that the structure of all information and relationships associated with the recipe specification of a pharmaceutical product could be formally represented. The classes in the recipe ontology were identified through discussions with domain experts, knowledge of their intended use and review of the literature on related topics.

2.4.2. Step 2: enumerate important terms in the ontology

Aside from identifying the various classes that must be defined, another important decision in designing ontologies is deciding which specific classes constitute a particular ontology. For example, on one extreme you could define all classes in one

Fig. 7. Define relation of UnitProcedure and Operation in OWL.

ontology, whereas on the other extreme you could have a separate ontology for each class. Having all classes in one ontology is advantageous because it allows you define relationships between properties of various concepts. However, it is against the general principle of a modular framework. Since a given concept can be used for very diverse set of applications, it is desirable to organize the ontologies such that each ontology consists of as few classes as possible, and then to use them hierarchically.

At the lowest level, each ontology can be treated as a building block. An ontology at the next level can be constructed by selecting the desired ontologies. For example, an ontology named Material contains a class named Material for defining properties of all pure chemical compounds. This class has one attribute for each physical property associated with a material. Some of the material attributes are name, chemical formula, molecular weight, state at standard temperature and pressure, density of standard state. The Material ontology can be used as a building block for other ontologies. For example, an ontology that defines material composition, named MaterialCompostion, contains three attributes named speciesName, compositionValue and measuringUnit. It imports the Material ontology. Similarly, the Stream ontology imports the MaterialComposition ontology.

We used Protégé (Version 3.1) (see http://protege.stanford.edu/), an ontology and knowledge-base editor, for creating ontologies. The key higher level concepts in the process ontology include Parameter, Material, Equipment, Reaction, RecipeElement under which Operation, UnitProcedure and Procedure are defined. Other important concepts include Port, FlowLink, Stream and FlowNetwork.

The Operation class has several subclasses. Each subclass has attributes or constraints to characterize the underlying physical/chemical change. For example, the Filter subclass has an attribute to let you specify the split fractions for each incoming specie. OWL provides the capability to describe complex relations between the concepts in the process information. For example, hasUnitProcedure property of Operation and hasOperations property of UnitProcedure are inverse functional relations. The hasOperation property of Port has cardinality restriction of 1. The inference engine can clarify the hierarchy of classes, and furthermore, check consistency to ensure the completeness and validity of the process information based on predefined consistency rules. As an example, a consistency rule is used to require that the values of hasUpstreamPort and hasDownstreamPort properties of a Stream be different.

2.4.3. Step 3: define the classes and class hierarchy

Class attributes define the properties of the objects in that class. The properties are of two types: object type and data type. The data type attribute values are of primitive types such as integer, float, string and so on. The object type attribute values are instance(s) of some other class(es). As a general principle, the semantic content of information increases when information is encapsulated in classes because explicit meaning can be associated with classes. For example, to completely define the melting point of a material you need three attributes, value type, value and measuring unit. In this case, value type would be temperature, and could be specified as a string or a unique identifier. The other way to accomplish this would be to define a class named Parameter which has subclasses for each value type such as temperature, press, mass and so on. A particular temperature would be an instance of the Temperature subclass, the Melting point would be specified as a suitable object of the Temperature subclass. Of course, the value of the melting point and the desired measuring unit are part of the instance of the object of Temperature subclass.

The latter way of defining a class and attributes has substantial impact in terms of semantics and data organization. For example, when all attributes of melting point are defined with the Material class the information is embedded in a Material class object. Therefore, the ability to interpret and process that information must be included in the implementation of the Material class. On the other hand, if melting point is defined as an instance of Temperature class, the entire object can be passed to the implementation of the Temperature class. More importantly, the implementation of the Temperature class can be universally accessed and kept up to date. If the attributes are defined every time they are used, then overheads of implementing and maintaining each occurrence increase significantly.

2.4.4. Step 4: create instances

The actual data associated with a class is generated through instances or objects of that class. For example, a specific chemical is an instance of the Material class. Since the Material class defines all physical and chemical properties of a material, the values of properties specific to that material are entered while creating its instance. The class definitions along with instances constitute the information repository. A unique URL is associated with instance(s).

2.5. Ontology and tools

In this project, the information is encoded using OWL. For tools to use this information, there are two main options. For tools that will be developed in the future, ontology can be used as a native repository to store declarative knowledge and the procedural knowledge can be encoded using the programming language in tools. For legacy tools with its own syntax or semantics to the information, the ontology can be used for importing and exporting information.

Several application programming interfaces are available for parsing, representation, database persistence and querying the ontology. These interfaces provide a unified approach for tools to access information repository (Knublauch, 2004). One of the most widely used libraries is Jena Semantic Web Framework (see http://jena.sourceforge.net/), in which OWL classes, properties and individuals are represented as generic Java classes like OntClass and Individual. However, there are some limitations in Jena. One of the most important is that there is no direct method to access all the properties of a given class, when the property has more than one domain. The Protégé-OWL programming interface Version 2.0, an extension of Jena, was used. This interface overcomes some of the limitations in Jena, provides higher level functionalities and makes it easier to access information in OWL.

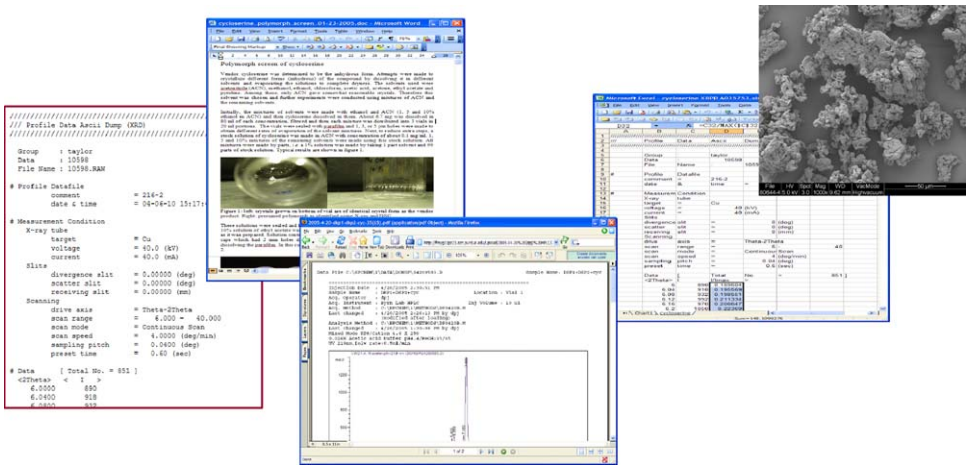

Fig. 8. Different formats of information in chemical product development.

2.6. Applications of the ontology

As a formal model of domain information, ontology is the foundation for various applications. In this work, we have demonstrated the use of ontology for information management and sharing, and discussed the benefits of ontology-driven software development.

2.6.1. Ontology-based experiment information management

Voluminous information is generated during product/process development, such as raw data generated from analytical instruments, pictures from SEM instrument, pictures for experimental settings, experimental notes and reports, mass and energy balance results from simulation tools, etc. The information could also be in different formats, including plain text files, WORD document, Excel worksheets, JPEG files, MPEG movies, PDFs, etc. (Fig. 8). How to effectively gather information from different resources, systematically manage the information, provide help in decision making become key tasks of information management.

Content management systems were explored to provide the backbone for the management of experimental data. The content management systems provide several data management functionalities that can be directly used in experimental information management, for example, user management, data upload, document management, etc. Most of the existing content or document management systems including e-LabNotebooks, Laboratory Information Management Systems or Content Management Systems, follow the file/folder structures similar to that in any operating system, in which folder hierarchies are created to store files. For each file, general and restricted sets of meta-data are usually applied, including the date when the file is created or modified, the person who created or modified the file, and description of the file. In order to find information, user may have to either manually go through the folder structure, or do a keyword-based search.

From our experience of implementing content/document management systems in our project, we found the following major limitations with the current structures:

- Meaning of the information not directly accessible. For example, the content of SEM pictures cannot be described only by file name.
- Lack of systematic way to represent and explore information relations. For example, relations between HPLC data distributed in various locations cannot be described.
- Domain specific information not easily supported. For example, quality assurance requires a standard for performing and reporting experiments; and interface for data entry needs to be created manually.
- The information in these systems cannot be easily shared with other software tools.

The file structure is usually created in an ad hoc way. For example, in the central storage for all experiment data, the folder structure is created based on person, time and purpose of a set of files it contains. The typical tree-like folder hierarchy cannot effectively represent the information relations which have indeed a graph structure.

The root cause of these problems is the lack of semantics of the information. A more flexible infrastructure which handles the semantics of the information in a systematic manner is required for information management. Given the semantic nature, ontology and Semantic Web related technologies would be the foundations for the new infrastructure. Meta-data can be defined and managed as ontology, and folder structure can be replaced with the ontology hierarchy. Research on developing Semantic Web Portal has demonstrated the potential of these techniques with very interesting prototypes (see http://www.ontoweb.org/).

In managing the information gathered from pre-formulation stages for pharmaceutical product formulation, we first created ontologies for material properties and experiments. The relations between *Material*, *Experiment* and *Properties* are defined in the ontology and represented graphically in Fig. 9. The central concept in this abstraction is the *material*, which represents pure substances and mixtures (which are characterized by pure substances and their compositions in all phases). A material has several properties (e.g. specific heat capacity), and can have sev-

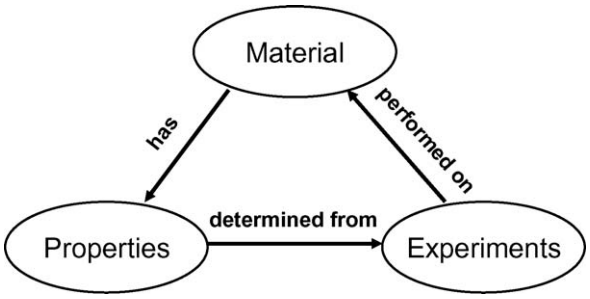

Fig. 9. Graph representation of relations.

eral experiments performed on it (e.g. compression tests) which determine the values of respective properties.

When formulation scientists perform an experiment and submit information to the central repository, an instance of corresponding ontology is created. Once the information model is created automatically from these instances, semantic search can be carried out. For example, consider the following search:

> "Find all the experiments that have been done on the micromeritics of Material A which are potentially affected by the relative humidity."

To carry out this search accurately, the following semantics are required: (1) the subclass relations in micrometritics; (2) the relations between experiment and material; and (3) the relations between context and experiment. Without the semantics, using a keyword-based search, many irrelevant documents would be returned.

We are further developing more natural ways to help scientists in creating the ontology instances and also investigating integration with common content management system like Opencms (see http://www.opencms.org/) and Mambo (see http://www.mamboserver.com/) to provide functionalities like user management, security and so on.

2.7. Ontology-based information sharing

As we demonstrate in this project, given its formal and semantic richness, ontology would completely define the related information and thus can be used to define a common vocabulary for information shared by the related tools.

Consider the following scenario: during safety analysis, the composition, temperature and pressure of material in equipment and streams are crucial. This information could be generated from simulation tools like Batches, which uses mathematical models for unit operations with material properties, or from manufacturing information systems for real process data if revalidation analysis is conducted. For a tool like PHASuite which assists the experts in safety analysis, without integration with simulation tools or manufacturing information systems, it has to develop its own simplified mass and energy balance calculations. The calculation might not be as accurate as provided by the simulation packages which deploy sophisticated unit operation models and physical properties calculations. This could limit the quality of the analysis.

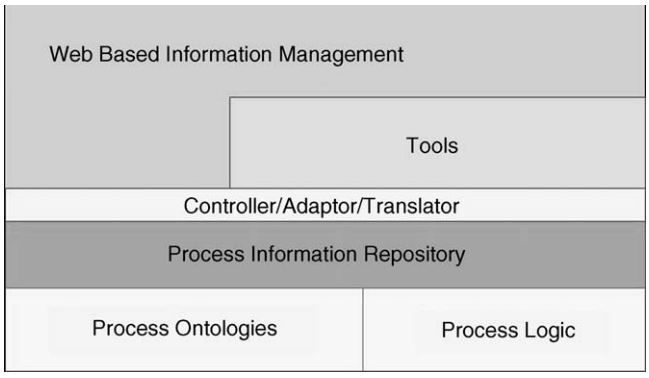

Fig. 10. Ontology-driven information management and sharing.

We proposed the ontology-driven information sharing in Zhao, Bhushan, & Venkatasubramanian (2003), and further demonstrated the methodology in Zhao, Joglekar, Jain, Venkatasubramanian, and Reklaitis (2005). In this demonstration, the material holdup and stream information calculated by Batches are stored in the information repository and can be directly used by PHASuite in conducting safety analysis.

Fig. 10 shows the proposed overall architecture of the API process information sharing and management. The implementation of this infrastructure can be divided into three main tasks. Process ontologies are first created as well as the logic embedded in the rules to ensure the completeness and validity of the process information. Based on the ontologies, instances of the concepts and relations are created for a particular process using the Web-based information management facility.

The second task is to create an interface to the repository so users can access, view and modify the information. A Web-based interface for information management was developed because the Web is the natural environment for the use of the infrastructure with such a wide scope. Additionally, development of thin-client applications in Web environment has become feasible due to the recent advances in Web technologies.

The third task is to provide application programming interface for various tools to access the information repository. The process information repository is in OWL format or related databases. Given the diversity of the tools that will access the process information, a middle layer consisting of controller, adaptor or translator is created for the tools as well as the Web interface to access the repository. From what we have shown, the process information can be sufficiently described using OWL, which can act as a process information repository. The open and explicit description of the information provides a foundation for sharing and integrating process information. Fig. 11 illustrates the process repository, and the interactions between the repository and tools, using a pharmaceutical API process as case study.

2.7.1. Methodologies for software development: ontology-driven and service oriented

This Semantic Web infrastructure is based on formal domain models in forms of ontologies that are linked to each other on

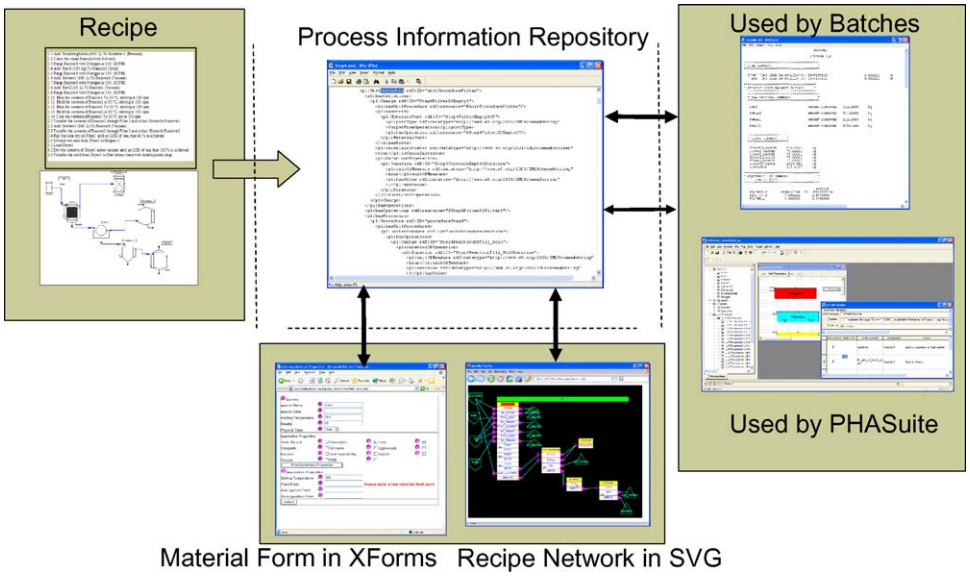

Fig. 11. Interactions between the process repository and application tools.

the Web. These linked ontologies provide the applications with shared terminologies and understanding. Based on these ontologies, an open infrastructure could be provided for intelligent agents and Web services.

The ontologies not only define the input/output data structure for software applications, they can also be used to represent the knowledge that the applications use to perform their tasks. The recent movement of Model-Driven Architecture (*MDA*) proposed by the OMG (Booch et al., 2004) explores ways to better integrate high-level domain models in Unified Modeling Language (UML) with software development. Ontology-driven approach is very similar to the MDA approach but in a much more extreme way (Knublauch, 2004): domain models are not only used for code generation, but they are used as executable artifacts at run-time. We are currently exploring these technologies to create a batch simulation platform for both quantitative and qualitative simulations and to better support the product/process development and lifecycle management.

Service-Oriented Architecture (SOA) promotes loose coupling between interacting software agents. Instead of a monolithic application package which tries to tightly control all heterogeneous parts, the modules which are loosely separable, should be constructed as services. SOA requires a small set of simple interfaces to all participating software agents, e.g. the Web Service Description Language 1.1 (WSDL specification, see http://www.w3.org/TR/wsdl). Descriptive messages express application-specific semantics, which is currently encoded in XML in most cases. OWL Web Ontology Language for Services (OWL-S) has been submitted to W3C as a markup language based on OWL for describing the properties and capabilities of Web services in an unambiguous, computer-interpretable form. OWL-S will facilitate better automation of tasks including Web service discovery, execution, composition and inter-operation (see http://www.w3.org/Submission/2004/07/).

3. Knowledge modeling and management for decision making

We consider knowledge as organized or contextualized information which can be used to produce new inferences and generate new information. All decisions in chemical product/process development are based on inferential knowledge, for example, solvent selection based on separation efficiency and safety/health/environmental concerns, equipment selection based on size requirements and material of construction, batch process schedules based on optimizing equipment utilization, etc. Among the various forms of knowledge, mathematical knowledge is the most concise, precise and abstract. Other forms of knowledge include heuristic knowledge which can be presented in forms of rules, and knowledge with the aim of guiding decisions which can be modeled as guidelines.

In this work, approaches for modeling the knowledge in the forms of guideline and mathematical knowledge have been proposed. These knowledge modeling efforts are based on the information modeling using ontology. These approaches are discussed using pharmaceutical product/process development as case study.

3.1. Modeling of guidelines

The process of pharmaceutical product development is a complex, iterative process consisting of selection of dosage form, excipients, processing routes, operating equipments and so on. Most of the decision making is carried out by domain experts based on the knowledge they possess. As the pharmaceutical product development is evolving from an art form to one that is more science and engineering based, there is increasing emphasis on explicit representation of the knowledge which goes into decision making. Knowledge Modeling helps in better understanding of the decision making process by making it more systematic.

Knowledge which goes into decision making in pharmaceutical product development can be classified into two categories—implicit and explicit knowledge. An example of implicit knowledge would be the knowledge which lies in the mind of an experienced domain expert. Explicit knowledge includes the formal decision trees and procedures a company may have or the guidelines from Food and Drug Administration (FDA) and International Conference on Harmonisation (ICH). However, as the amount of knowledge and information increase, it becomes very difficult for domain experts to use this knowledge. For effective use, this knowledge must be available in explicit form which can be directly understood by computers.

Traditional ways to model knowledge in computer interpretable format are either programming based or rule based. In the programming-based approach, all the logic is encoded in a computer program and therefore is not transparent. A user cannot change the logic without accessing the source code, which sometime may not be open or the user may not have the understanding of particular programming language. Most of the expert systems use the rule-based approach. Rules are effective in capturing the knowledge but are not flexible. Most of the times, rules are not structured properly, not scalable and difficult to maintain. Also, it is difficult to predict how different rules interact with each other.

3.1.1. Structure of guideline

We use ontology-based approach to model a guideline. Guidelines are created based on the GuideLine Interchange Format (GLIF) (Peleg et al., 2004) Ontology, which is a specification developed mainly for structured representation of clinical guidelines. GLIF was developed by the InterMed Collaboratory to facilitate sharing of clinical guidelines. The InterMed collaboratory is a joint project of medical informatics laboratories at Harvard, Stanford, Columbia and McGill Universities. GLIF supports computer-based guideline execution. It provides a representation for guidelines which is computer interpretable as well as human readable. It is independent of computing platforms which enables sharing of guidelines.

In the GLIF specification, a guideline is represented as an instance of the *guideline* class. Each guideline has the *intention* attribute which defines the purpose of guideline as text. The process of decision making is encoded as the *algorithm* of a guideline. Within an algorithm, instances of five types of tasks, which are called *guideline steps*, can be encoded and linked together in a flowchart to specify their scheduling and coordination during guideline application. Specifically, *action steps* are used to record clinical or computational actions; *decision steps* are used to represent decision points; *patient state steps* are used to specify a patient's states in the specific contexts of a guideline's application; and *branch steps* and *synchronization steps* are used to schedule and coordinate concurrent tasks. The clinical care process represented in the GLIF model can be nested using *subguidelines*, thus multiple views to the care process with different granularities can be defined. A detailed summary of GLIF can be found in Peleg et al. (2004).

3.1.2. Guideline development

In this work, various guidelines were developed for the pharmaceutical product development based on the knowledge collected from detailed discussions with the academic collaborators at Department of Industrial and Physical Pharmacy, Purdue University. As a case study, guidelines were used for the development of a drug product with Cycloserine as the API, to treat the Multidrug-Resistant Tuberculosis (MDRTB). Some details of how different guidelines are modeled and used are discussed here.

Every guideline has a specific purpose and the details of a guideline are specified in algorithm as shown in Fig. 12. Algorithm consists of the instances of five types of steps linked together to form a flowchart. Fig. 12 shows the guideline for "Drug Product Development" with the purpose "Given the API and dose, drug product development (selection of dosage form, processing route and excipients)."

State steps define the starting and end points of a guideline. State step points to the development state instance which contains the information of existing state of pharmaceutical product development. To start using the guideline, first an instance of *development state* is created. To populate the development state instance, the API for which the pharmaceutical product development is desired is selected from the list of materials from material ontology. Other details which are known at this stage are also added, for example, dose.

The action step *Development_for_IR_Solid_Oral_Dosage_Form* in the *Drug_Product_Development* guideline (Fig. 12) invokes the subguideline, shown in Fig. 13, which selects processing routes and excipients. Guideline in Fig. 13 has examples of various guideline steps. Branch step (Start_Excipient_Selection_for_Various_Selected_Routes) and synchronization step (End_Excipient_Selection_for_Various_Selected_Routes) are used for simultaneous execution of steps. Action step is used for doing computations, giving specific recommendations, and starting a subguideline. For example, there are subguidelines for the initial selection of processing routes based on physical, chemical and mechanical properties of the API.

Every decision criterion is specified using decision step. The decision criterion can be changed based on the requirements. For example, the criterion 'Is Dose Low' can be relaxed from "dose < 100 mg" to "dose < 200 mg." The value of the property used in a decision criterion is directly linked to the list of properties. Decision step also specifies the option to select based on the outcome of a decision.

3.1.3. Execution of guideline

For clinical guidelines, a Guideline Execution Engine (GLEE) has been developed to interpret guidelines encoded in the GLIF format and to integrate with clinical information systems for guideline implementation. A detailed summary of GLEE can be found at Wang et al. (2004). In order to provide the decision support for drug product development a Java based execution engine was developed to execute guidelines. It is linked to guideline and other ontologies.

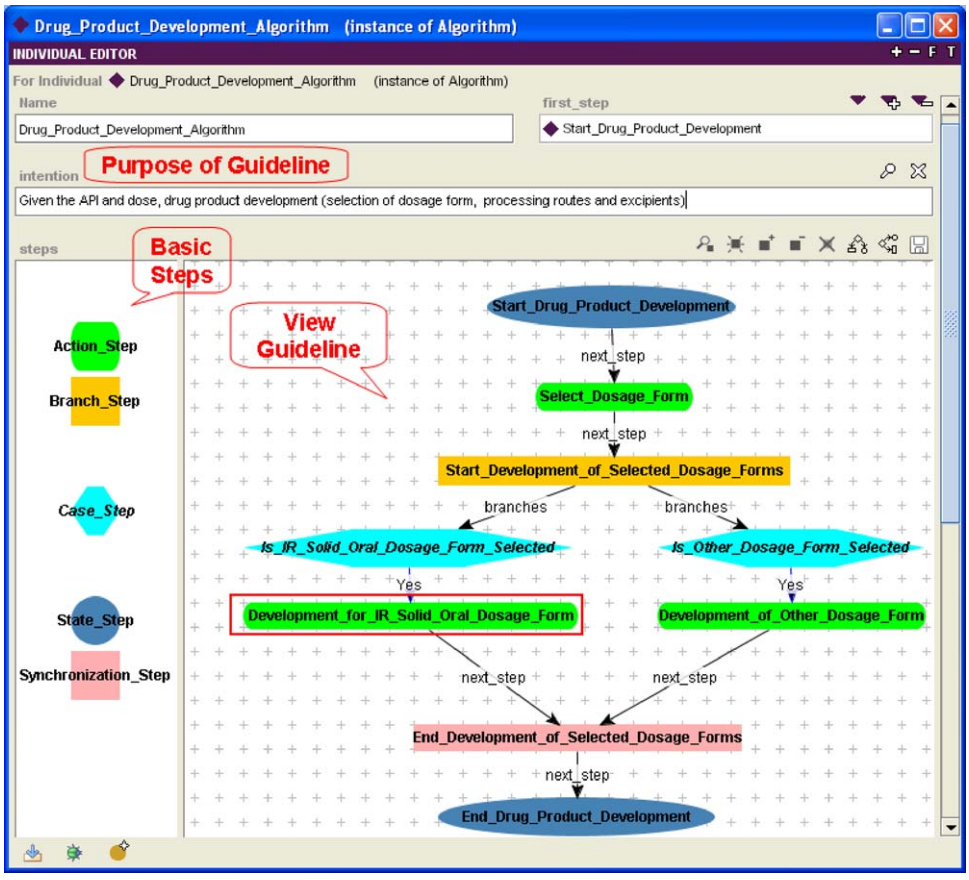

Fig. 12. Basic steps in a guideline algorithm.

As compared to human experts, knowledge in the form of guidelines is permanent and transferable. In a large search space, decision making is faster and more consistent. GLIF enables the modeling and representation of guidelines in a structured manner. Guidelines effectively capture the knowledge and can be used for modeling the knowledge in different domains in chemical and pharmaceutical engineering and in development of decision support systems. Apart from helping domain experts in decision making, guidelines can also help them in better understanding of development process as well as in training.

3.2. Managing mathematical knowledge

Large amount of knowledge used in chemical product and process development is in the form of mathematical equations, which is referred to as mathematical knowledge. Compared to other forms of knowledge, like rules, guidelines, etc., mathematical knowledge is more abstract and highly structured (Farmer, 2004).

As an example, the equations and variables for fluid-bed drying model (Kunii and Levenspiel, 1969) are shown in Fig. 14. In this model, the drying process is divided into two stages; the first stage, modeled by the first two equations, is the vaporization of moisture on the surface, while the second stage, modeled by the third equation, is the diffusion of water out of the particles. These equations depend on material properties, equipment parameters and operating conditions.

In pharmaceutical process development, most of the mathematical knowledge is either embedded in specific software tools such as unit operation models in a simulation software, or has to be entered into a more general mathematical tool following a specific syntax, such as Matlab or Mathematica. Much of this knowledge, however, concerns specific applications and is expressed procedurally rather than declaratively. For example, in the application domain of chemical process development, Aspen Custom Modeler (AspenTech, 2005) provides a modeling language for creating new models which can be used with other Aspen products. Typically, writing and editing of programming language-based models is difficult because it requires familiarity with the syntax of the particular solver used (e.g. Matlab, Mathematica or DASSL).

The information technologies are transforming how mathematical knowledge is represented, communicated and applied. Mathematical Knowledge Management, a new interdisciplinary field of research, has attracted researchers from mathematics, computer science, library science and scientific publishing. Marchiori (2003) provides a general account of technologies like XML, RDF and OWL to foster the integration of Mathematical representation and Semantic Web. By doing so, it becomes possible to integrate various mathematical sources, to search globally, to associate with meta-data as context, and to integrate with other forms of knowledge. Caprotti, Dewar, and Turi (2004) discussed a mechanism for encoding information on mathematical Web services to identify automatically the requirements for

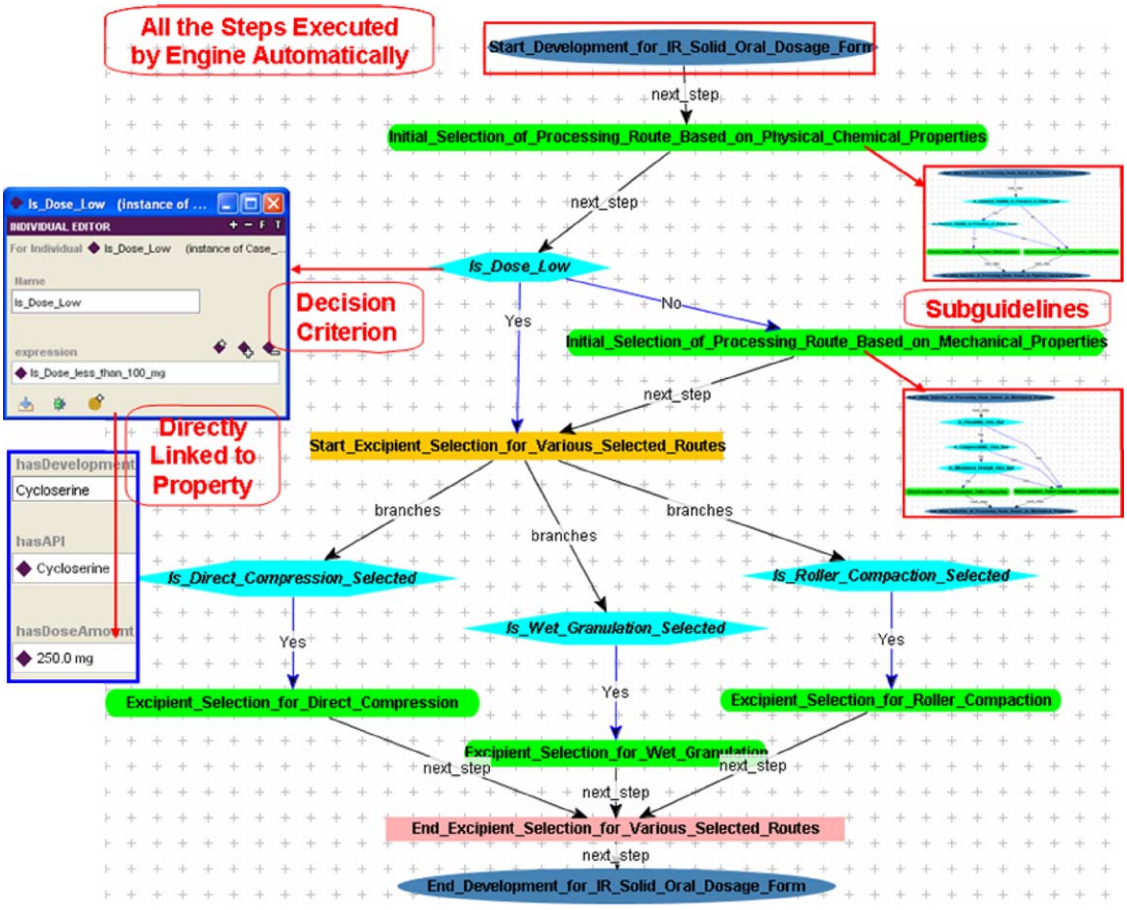

Fig. 13. A guideline showing the details of various steps.

performing a particular task. This mechanism utilizes OWL and Description Logic reasoning capability.

Bogusch and Marquardt, 1997 refer to an ontology-based approach for managing process model equations that defines the semantics between the equation and the associated variables. Such an object-oriented approach for modeling mathematical knowledge based on behavioral decomposition is very fundamental and therefore reusable. Although many research groups have tried to use such an ontology based framework for managing process models, for example, ModKit (Bogusch et al., 2001), ProMot (Mangold et al., 2005), etc. Most of them rely upon the simulation environment/engine to analyze and solve the systems of equations instead of using powerful dedicated solvers which are available. Also, they use symbolic representation for

$$M_t = M_o - kt$$

$$k = \frac{\rho_g}{\rho_s} \cdot \frac{C_{pg}\,(T_{\text{inlet}} - T_{\text{exit}})}{H_v} \cdot \frac{U_o}{(1-\varepsilon_m)L_m}$$

$$M_t = \frac{6}{\pi^2}\left[\sum_{n=1}^{\infty}\frac{1}{n^2}\exp\left(-n\pi^2\frac{D_m t}{R^2}\right)\right](M_o' - M_\infty) + M_\infty$$

where,

M_0 is the initial moisture content	M_t is the moisture content at time t
ρ_g is the density of gas	ρ_s is the density of solid
C_{pg} is the specific heat of gas	L is the latent heat of vaporization
L_m is the height of the fixed bed	T_e is the temperature of gas at exit of the bed
ε_m is the void fraction of the bed	T_{gi} is the temperature of gas at inlet of the bed
U_0 is the superficial gas velocity	M_0' is the moisture content at the end of linear phase
D_m is the solvent diffusivity	M_∞ is the desired end moisture content
R is the average radius of particles	

Fig. 14. Model for fluid-bed drying.

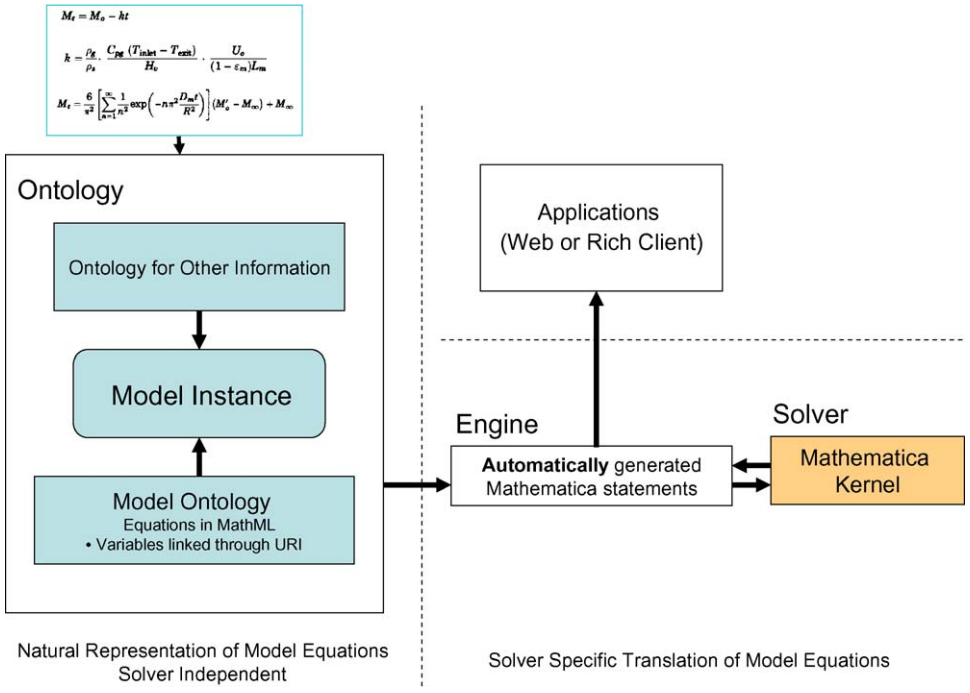

Fig. 15. The proposed approach for modeling mathematical knowledge.

equations and variables which then is parsed into a format that the simulation engine can understand (Mangold et al., 2005). The expensive license fee for most of these model development softwares is also a point of concern. The components of the simulation environments discussed above are specialized and difficult to adapt to other applications.

In the proposed approach, the declarative and procedural parts of the mathematical knowledge have been separated. The declarative part consists of the information required by the model to run, the information generated from the model, and the model equations.

The declarative part consists of the *ModelDefenition* ontology and an information repository, also an ontology. The model ontology consists of three classes named DependentVariables, IndependentVariables and ModelParameters that are used to define the dependent variables, independent variables, and model parameters, respectively. Another class, named ModelEquations, is used to define all model equations. Each equation is stored as Content MathML, a W3C recommended standard based on XML.

The model ontology is written in OWL using Protégé. Several Web-based graphical editors , such as WebEq (see http://www.dessci.com/en/products/webeq/), are available to create mathematical equations and store them in Content MathML. Thus, the process of creating a mathematical model becomes very intuitive and user friendly compared to the existing approaches. Each model is an instance of the *Model* class of the ontology. For example, the fluid-bed drying model in the ontology calculates the moisture content in the output of the dryer as a function of time. The two equations, one for the heat transfer controlled regime and one for mass transfer controlled regime are shown in Fig. 15. A model ontology allows representing mathematical equations naturally in a form that is independent of the solver. The solution of the model equation is governed by the context in which that model is used. For example, the drying model can be used in a stand alone mode to predict the moisture profile in a specific dryer under given conditions which are retrieved from the information repository.

In this work, models consisting of algebraic and first order explicit differential equations describing simple unit operations were created and Mathematica was used as the solver. Mathematica has several useful features, including: (1) the symbolic processing capability which handles equations in MathML formats directly, without having to translate into procedures as in other general mathematic packages; (2) the extensibility with programming languages like Java, which makes it possible to communicate between the Mathematica kernel and the engine; (3) the Web capability which allows the use of Web environment to access mathematica installed on remote machines.

The procedural part of this approach is implemented using a Java-based engine. It constructs Mathematica commands based on equations in MathML. It also creates statements to initialize the model parameters with values provided in the instance of the model class chosen, and invokes the Mathematica kernel to solve the set of equations. A graphical user interface (GUI) is used to display results from the solver (plots or expressions) and is used to select the instance of the model to be solved.

Due to its modularity, the proposed information centric approach allows clear separation between model creation and solution. This approach provides a systematic way for model creators to describe the models in terms of equations with the help of the intuitive and visual equation editor, and the variables which are described using the ontology and linked to the

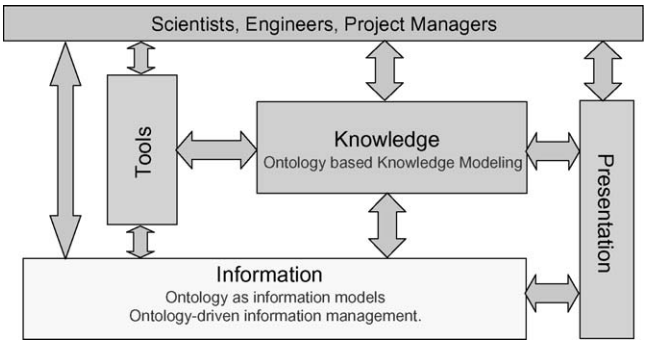

Fig. 16. The ontology-driven informatics infrastructure.

information resources. The MathML description of the equations and the ontologies provide an open and solid foundation for the engine to understand the equations and variables, along with the links to access the values of the variables during execution of the model. The equation solving itself is performed by Mathematica. User friendly, Web-based or thick-client interfaces can be easily created for the use of existing models. By creating solver specific engines, a range of solvers can be made available to the users.

The overall structure of the proposed ontology-driven informatics infrastructure is shown in Fig. 16. The proposed approach clearly delineates information, knowledge, software tools and presentation blocks used in pharmaceutical product development. The Presentation layer encompasses all interactions between the end user and the functional components of the infrastructure, for example, browsing or uploading information in the repository, invoking analysis tools, guideline execution and so on. The bidirectional arrows indicate two-way exchange of information between the connecting blocks.

4. Conclusions

In this paper, we propose an ontological information-centric infrastructure to support product and process development in the pharmaceutical manufacturing domain. An ontology is used to model the information, while Semantic Web provides a general framework for implementing the infrastructure. Various ontology languages and prior attempts in other chemical engineering domains have been reviewed. Process recipe information is used as an example to demonstrate the major steps in developing a domain specific ontology. Ontology-based information management and information sharing are discussed. Modeling of two different forms of knowledge, namely, guideline and mathematical knowledge was discussed.

An ontological framework can better support chemical product and process development and also open new opportunities for applications including product/process lifecycle management, development history management and so on. The ontological informatics infrastructure proposed in this paper is at the dawn of a new paradigm for representing, analyzing, interpreting and managing large amounts of complex and varied information for product development and manufacturing. Considerable intellectual and implementation challenges lay ahead but the potential rewards will completely transform how we do product development and manufacturing in the future.

Acknowledgments

The authors gratefully acknowledge the support from The Indiana 21st Century Research and Technology Fund. Thanks also go to our other collaborators at Department of Industrial and Physical Pharmacy at Purdue University, Information Technology at Purdue (ITaP) and Eli Lilly and Company.

References

Aspen Tech (2005). *Aspen Custom Modeler*. http://www.aspentech.com/product.cfm?ProductID=54.

Antoniou, G., & Harmelen, F. V. (2004). *A Semantic Web primer*. Cambridge, MA: The MIT Press.

Bañares-Alcántara, R., & Lababidi, H. M. S. (1995). Design support systems for process engineering—II: KBDS: An experimental prototype. *Computers and Chemical Engineering*, *19*(3), 279–301.

Batch Process Technologies Inc. (2005). *Batches reference manual*. W. Lafayette, IN.

Booch, G., Brown, A., Iyengar, S., Rumbaugh, J., & Selic, B. (2004). An MDA manifesto. *MDA Journal*, *5*, 2–9.

Branchman, R. J., & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, *9*(2), 171–216.

Bogusch, R., & Marquardt, W. (1997). A formal representation of process model equations. *Computers & Chemical Engineering*, *21*(10), 1105–1115.

Bogusch, R., Lohmann, B., & Marquardt, W. (2001). Computer-aided process modeling with ModKit. *Computers & Chemical Engineering*, *25*(7–8), 963–995.

Burkett, W. C., & Yang, Y. (1995). The STEP integration information architecture. *Engineering with Computers*, *11*, 136–144.

Caprotti, O., Dewar, M., & Turi, D. (2004). Mathematical service matching using description logic and OWL. In *Proceedings of MKM 2004*.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice J. P. (1998). *Open knowledge base connectivity 2.0.3*. Technical report.

Farmer, W. M. (2004). MKM: A new interdisciplinary field of research. *ACM SIGSAM Bulletin*, *38*(2).

Farquhar, A., Fikes, R., & Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human Computer Studies*, *46*(6), 707–727.

FDA (2005). *Process analytical technology initiative*. http://www.fda.gov/cder/OPS/PAT.htm.

Gruber, T. R. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition*, *5*(2), 199–220.

Jarke, M., & Marquardt, W. (1996). Design and evaluation of computer-aided process modeling tools. In J. F. Davis, G. Stephanopoulos, & V. Venkatasubramanian (Eds.), *Intelligent systems in process engineering, AIChE symposium series no. 312, 92*s, 97–109.

Killich, S., Luczak, H., Schlick, C., Weissenbach, M., Wiedenmaier, S., & Ziegler, J. (1999). Task modeling for cooperative work. *Behavior and Information Technology*, *18*, 325–338.

Knublauch, H. (2004). Ontology-driven software development in the context of the Semantic Web: An example scenario with Protégé/OWL. In *International workshop on the model-driven Semantic Web*.

Kunii, D., & Levenspiel, O. (1969). *Fluidization Engineering*. New York: John Wiley & Sons.

Lenat, D. B., & Guha, R. V. (1990). *Building large knowledge-based systems: Representation and interface in the Cyc project*. Boston, MA: Addison-Wesley.

MacGregor, R. (1991). Inside the LOOM classifier. *SIGART Bulletin*, *2*(3), 70–76.

Mangold, M., Angeles-Palacios, O., Ginkel, M., Kremling, A., Waschler, R., Kienle, A., & Gilles, E. D. (2005). Computer-aided modeling of chemical and biological systems: Methods, tools, and applications. *Industrial & Engineering Chemistry Research*, *44*(8), 2579–2591.

Marchiori, M. (2003). The mathematical Semantic Web. In *Proceedings of the international conference on mathematical knowledge management*.

McGreavy, C., Wang, X. Z., Lu, M. L., & Naka, Y. (1995). A concurrent engineering environment for chemical engineering. *Concurrent Engineering: Research and Applications*, *3*(4), 281–292.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05.

OWL (2004). *Web ontology language overview. W3C Recommendation*. http://www.w3.org/TR/owl-features/.

Paszko, C., & Pugsley, C. (2000). Considerations in selecting a Laboratory Information Management System (LIMS). *American Laboratory*, *9*, 38–42.

Peleg, M., Boxwala, A., Tu, S., Wang, D., Ogunyemi, O., & Zeng, Q. (2004). *Guideline Interchange Format 3.5 Technical Specification*. http://smi-web.stanford.edu/projects/intermed-web/guidelines/.

Schneider, R., & Marquardt, W. (2002). Information technology support in the chemical process life cycle. *Chemical Engineering Science*, *57*, 1763–1792.

Sistare, F., Berry, L. P., & Mojica, C. A. (2005). Process analytical technology: An investment in process knowledge. *Organic Process Research and Development*, *9*(3), 332–336.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *IEEE Transactions on Data and Knowledge Engineering*, *25*(1–2), 161–197.

TopQuadrant (2003). Semantic integration: Strategies and tools. *TopQuadrant technology briefing*.

Wang, D., Peleg, M., Tu, S. W., Boxwala, A. A., Ogunyemi, O., Zeng, Q., et al. (2004). Design and implementation of the GLIF3 guideline execution engine. *Journal of Biomedical Information*, *37*(5), 305.

Zall, M. (2001). The Nascent Paperless Laboratory. *Chemical Innovation*, *31*, 2–9.

Zhao, C. (2002). *Knowledge engineering framework for automated HAZOP analysis*. Ph.D. Thesis. Purdue University.

Zhao, C., Bhushan, M., & Venkatasubramanian, V. (2003). Roles of ontology in automated process safety analysis. In *Proceedings of ESCAPE 13*.

Zhao, C., Joglekar, G., Jain, A., Venkatasubramanian, V., & Reklaitis, G. V. (2005). Pharmaceutical informatics: A novel paradigm for pharmaceutical product development and manufacture. In *Proceedings of ESCAPE15*.