Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2013), *27*, 65–82.
© Cambridge University Press 2013 0890-0604/13 $25.00
doi:10.1017/S0890060412000352

PRACTICUM ARTICLE

QuenchML: A semantics-preserving markup language for knowledge representation in quenching

APARNA S. VARDE,[1] MOHAMMED MANIRUZZAMAN,[2] AND RICHARD D. SISSON, JR.[3]

[1]Department of Computer Science, Montclair State University, Montclair, New Jersey, USA
[2]Center for Heat Treating Excellence, Metal Processing Institute, Worcester Polytechnic Institute, Worcester, Massachusetts, USA
[3]Department of Manufacturing and Materials Engineering, Worcester Polytechnic Institute, Worcester, Massachusetts, USA

(Received March 27, 2010; Accepted October 22, 2011)

Abstract

Knowledge representation (KR) is an important area in artificial intelligence (AI) and is often related to specific domains. The representation of knowledge in domain-specific contexts makes it desirable to capture semantics as domain experts would. This motivates the development of semantics-preserving standards for KR within the given domain. In addition to the storage and analysis of information using such standards, the effect of globalization today necessitates the publishing of information on the Web. Thus, it is advisable to use formats that make the information easily publishable and accessible while developing KR standards. In this article, we propose such a standard called Quenching Markup Language (QuenchML). This follows the syntax of the eXtensible Markup Language and captures the semantics of the quenching domain within the heat treating of materials. We describe the development of QuenchML, a multidisciplinary effort spanning the realms of AI, database management, and materials science, considering various aspects such as ontology, data modeling, and domain-specific constraints. We also explain the usefulness of QuenchML in semantics-preserving information retrieval and in text mining guided by domain knowledge. Furthermore, we outline the significance of this work in software tools within the field of AI.

Keywords: Domain Expertise; Heat Treating of Materials; Information Retrieval; Semantic Web; Text Mining

1. INTRODUCTION

Knowledge representation (KR) in scientific domains often requires capturing domain expertise. Increasing globalization in recent years makes it imperative to provide up-to-date information retrieval (IR) on the Web. These factors lead to the popularity of the eXtensible Markup Language (XML; World Wide Web Consortium, 2004*c*), a Web publishing standard with semantic tags helping to capture domain knowledge. In our work, the domain of focus is *quenching* in the realm of materials science. Quenching involves rapid cooling operations during heat treating of materials to achieve desired mechanical and thermal properties (Totten et al., 1993). We propose the Quenching Markup Language (QuenchML) as a KR standard for semantics-preserving storage of quenching information.

In this article, we provide the motivation for proposing QuenchML, giving an overview of the more general Materials Markup Language (MatML; Begley, 2003). We describe the development of QuenchML, encompassing data modeling with entities and relationships (Chen, 1976), ontology definition (Russell & Norvig, 2009), and schema design (World Wide Web Consortium, 2004*c*). We explain the use of Semantic Web standards such as the Web Ontology Language (OWL; World Wide Web Consortium, 2004*a*) in QuenchML development and the deployment of XML schema constraints (World Wide Web Consortium, 2004*c*). We also explain the functionality of QuenchML in semantics-preserving data storage and IR, considering declarative forms of access such as XQuery (Boag et al., 2003). We emphasize the use of QuenchML in data mining and machine learning, especially in mining over plain text sources where it simulates the role of a domain expert to discover meaningful knowledge and does so efficiently, saving human time and effort. We exemplify this through a system called RuleExtractor (Varde, Aker, et al., 2009) that uses QuenchML for text mining, along with machine learning tools such as the Waikato Environment for Knowledge Analysis (WEKA; Witten & Frank, 2005) and natural language processing (NLP) tools such as the Stanford Parser (Klein & Manning, 2003). We further discuss artificial intelligence (AI) software development that would benefit from this work (e.g., building and enhancing expert systems; Varde et al., 2003).

Reprint requests to: Aparna S. Varde, Department of Computer Science, Montclair State University, 1 Normal Avenue, Montclair, NJ 07043, USA. E-mail: vardea@montclair.edu

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

In short, we make the following main contributions in this article:

- propose a KR standard in quenching called QuenchML;
- illustrate the use of XML features and Semantic Web standards in QuenchML;
- describe the use of QuenchML in IR, along with MatML;
- emphasize the significance of QuenchML in text mining; and
- discuss the development of AI software that can benefit from QuenchML.

This article will be of interest to the KR and Semantic Web communities, data mining/machine learning professionals, and materials science users.

The rest of this article is organized as follows. Section 2 gives the background and motivation. Section 3 describes QuenchML development, and Section 4 addresses the issue of constraints. Section 5 explains QuenchML use in KR. Section 6 outlines the role of QuenchML in data mining, and Section 7 provides a discussion on using QuenchML in AI software tools. Finally, Section 8 gives the summary and conclusions.

2. BACKGROUND AND MOTIVATION

2.1. KR in AI

KR is an area of AI that creates formalisms to depict concepts ranging from general to specific (Russell & Norvig, 2009). It could be *pragmatic*, which involves general world knowledge, or *semantic*, which involves meaning with reference to context often in one or more domains.

A classical KR formalism is first-order logic. This allows us to state facts about categories, using certain symbols. For example, $\in$ means *element of*, $\supset$ means *superset*, $\Rightarrow$ means *implies*, and $\wedge$ means *and*, as shown in the following sample representation (Russell & Norvig, 2009):

1. One category is a superclass of another (e.g., Animal is a superclass of Zebra):

$$\text{Animal} \supset \text{Zebra}$$

2. A given object is an element of a category (e.g., object Z is a Zebra):

$$Z \in \text{Zebra}$$

3. An object has certain properties if it belongs to a category (if Z is a Zebra, it implies that Z has black stripes and white stripes):

$$Z \in \text{Zebra} \Rightarrow \text{BlackStripes}\ (Z) \wedge \text{WhiteStripes}\ (Z)$$

Such knowledge, although intuitive to humans, needs to be fed into a computer to process information. In the context of specific domains, there is a need to formally store facts that constitute domain knowledge. First-order logic serves as a classical means to provide this formalism and has been traditionally used in AI.

KR has been further discussed in the interdisciplinary literature in AI and engineering. The issue of representing knowledge specifically for applications pertaining to the Semantic Web has been addressed by Felfering et al. (2003). The use of ontology in IR has been emphasized in works such as Li and Ramani (2007). Cross-domain IR is described in the work of Chiu and Shu (2007). They propose the specific concept of biomimetic design considering NLP. First-order logic has been further used in several works. An interesting piece of work involves enhancing knowledge management by the use of first-order logic with specific reference to engineering design by Witherell et al. (2009). Likewise, other standards have been developed and used in the AI and engineering communities.

Other relevant literature in KR includes work in Thomere et al. (2002), where a system of nomenclature is developed with the goal of enabling domain experts to build knowledge bases without relying on human scientists. Another interesting piece of work is described in Riloff (1996), where domain-specific dictionaries are created for information extraction given an appropriate training corpus. Miller et al. (1990) proposed an online lexical database that attaches different possible meanings to words in several contexts, making an important contribution to KR that is globally accessible. Hence, the vast realm of KR has been addressed by several researchers from various perspectives.

Today, XML and Semantic Web standards such as the OWL (World Wide Web Consortium, 2004*a*) can serve as means for KR. They are found highly suitable in the context of globalization and also help to capture semantics, so they are very useful for storing information in science and engineering domains. We shall discuss these in more detail with specific reference to our work. Before that, we present a brief background on our domain of focus, quenching.

2.2. The quenching domain

Heat treating is an area of materials science that deals with the combination of operations involving controlled heating and cooling of a metal in the solid state for the purpose of obtaining certain mechanical and thermal properties. Quenching involves rapid cooling of a material to make it undergo transformations and is a very crucial step in heat treatment, thereby forming an important branch of study in the heat treating of materials (Totten et al., 1993; Mills, 1995).

The material undergoing quenching is normally referred to as the *part*, *probe*, or *workpiece*. The cooling medium is known as the *quenchant*. Quenchants can be of various types, such as oil, water, gas, and polymer. These have properties, including age (how old it is), viscosity (its capacity to flow), and degradation (how much it loses its quality over time). The part is made of a certain alloy. It has characteristics, namely, geometry (shape of the part), surface roughness (rough, smooth, or medium), oxidation (formation and its

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

thickness), size (dimensions of the part), and more. The part also has some details pertaining to its manufacturing. Some of these are the occurrence of welding (fabrication by joining) and stamping (whether the part was stamped during production). There are also properties based on the microstructure of the alloy (e.g., uniformity of the part grains).

An example of a quenching process is illustrated in Figure 1, which shows a typical apparatus used for experiments at the Center for Heat Treating Excellence (CHTE) at Worcester Polytechnic Institute (WPI). During the quenching process, the quenchant has a certain temperature and a certain level of agitation (extent to which the fluid is stirred). The part has a certain orientation when it is suspended into the quenchant (e.g., it could be immersed vertically or horizontally). The quench conditions affect the rate of cooling (fast, slow, or medium), the nature of cooling (uniform or nonuniform), and other factors.

After quenching, the part acquires desired properties (e.g., a specific level of hardness). There is also a tendency for distortion in the part while being quenched, that is, the part gets deformed to some extent. It is obviously desirable to prevent distortion or to minimize its extent (Totten et al., 1993). There are factors affecting distortion, including cooling rate, cooling uniformity, nature of the quenchant, and the alloy used in the part.

Data gathered during quenching has traditionally been stored in sources such as flat files and more recently is being stored in database formats. However, there could be some raw data scattered in different locations. There is related literature in quenching such as research papers, user specifications, technical reports, and other text sources. This is likely to be stored in bibliographic databases or as hard copies of documents. This is mostly plain text along with diagrams, graphs, and numeric values. In order to publish all this information in a meaningful manner and facilitate its exchange, there is a need for an efficient medium of communication that preserves the semantics of the quenching process. There is also a need to have all the data in a common format for easy access.

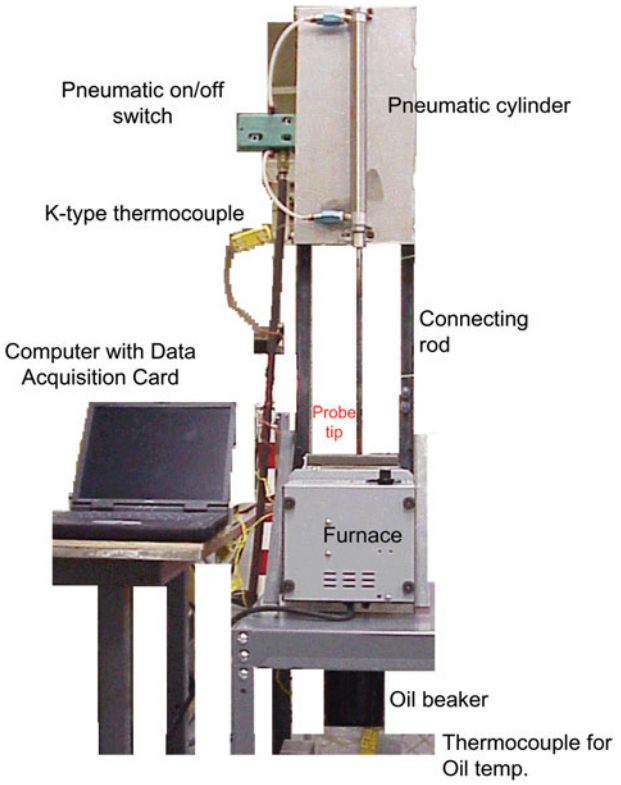

Fig. 1. The Center for Heat Treating Excellence quenching apparatus. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

2.3. Hyper Text Markup Language (HTML) and XML

HTML is the programming language that Internet browsers typically use to display a Web page. Each element, commonly known as a *tag*, contains an instruction commanding the browser how to display images and words (Bouvier, 1995). An example of HTML data storage appears in Figure 2. This is a very simple example referring to a bookstore. The tags such as <*HEAD*> and <*BODY*> are called *elements*, and their properties such as *bgcolor* and *alink* are called *attributes.*

HTML has its limitations. Its tags do not capture semantics. They stress more the display and form of the information than its content. For example, in Figure 2, the link would appear in red, a detail of presentation *not semantics*. This poses problems in interpretation with reference to context. In addition, HTML has a fixed tag set, not extensible to a domain. Another issue is that the data stored in other formats such as relational databases needs conversion to be stored in HTML. Because it is essential to have up-to-date information available on the Web, conversion needs to be performed whenever new data have to be published. This is a recurrent and time-consuming task.

XML is designed to improve the functionality of the Web by providing more flexible and adaptable information identification. It is called extensible because it is not a fixed format like HTML (a single, predefined markup language). XML is a metalanguage, that is, a language for describing other languages that facilitates designing customized markup languages for different types of documents (Flynn, 2002).

Figure 3 shows an example of data storage in a bookstore using XML. Here, *elements* are tags such as <*book*> and

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
 <TITLE>BookStore</TITLE>
</HEAD>
<BODY bgcolor="white" text="black"
  link="red" alink="blue" vlink="maroon">
  ... document body...
</BODY>
</HTML>
```

Fig. 2. A Hyper Text Markup Language example.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

```
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

Fig. 3. An eXtensible Markup Language example.

<*title*>, and *attributes* are properties such as *category* and *lang* (language). Thus, elements and attributes are descriptive with respect to the domain. The format of XML is described as semi-structured, because it is not fully structured like relational data nor is it unstructured like free-flowing natural language. It seamlessly allows storage of textual and nontextual formats in a common integrated fashion. For example, experimental numerical data and related plain text literature in a given domain can be stored in a common XML document.

Due to the extensible nature of XML, there are domain-specific markup languages defined within its context. A few examples include Chemical Markup Language (Murray-Rust, 1997), Mathematics Markup Language (Carlisle et al., 2001), and Medical Markup Language (Guo et al., 2003). These follow the XML syntax and encompass the semantics of the concerned domain. We discuss one such language, MatML, because it is relevant to our work.

2.4. MatML for materials science

The domain-specific markup language MatML was developed to serve as the XML for materials property data. This was proposed by the National Institute of Standards and Technology (Begley, 2003). It has been used in applications where domain semantics is important (e.g., Fahrenholz, 2006; Varde et al., 2006). Figure 4 is an example of data stored using MatML (Begley, 2003).

This example depicts the storage of aluminum alloy data. It can be seen from this example that compared to HTML tags and attributes that only explain presentation details, MatML is more meaningful. A few MatML tags shown here are <*Material*>, <*BulkDetails*>, and <*PropertyData*>. These store information about the material, its details in bulk as a whole, and data on its specific properties.

```
<MatML_Doc>
  <Material>
    <BulkDetails>
      <Name>1350</Name>
      <Class>metal</Class>
      <Subclass>aluminum alloy</Subclass>
      <Specification>ASTM B230</Specification>
      <ProcessingDetails>
        <Name>H18</Name>
      </ProcessingDetails>
      <PropertyData property="p1" source="s1">
        <Data format="integer">23,17,15</Data>
      </PropertyData>
    </BulkDetails>
    <Metadata>
      <DataSourceDetails id="s1">
        <Name>
        "Properties of Aluminum Alloys - Tensile,
        Creep, and Fatigue Data at High and Low
        Temperatures."
        </Name>
      </DataSourceDetails>
      <PropertyDetails id="p1">
        <Name> Axial-Stress Fatigue Strength</Name>
        <Units name="ksi" Description="kip per square inch">
          <Unit>kip</Unit>
          <Unit power="-2">inch</Unit>
        </Units>
      </PropertyDetails>
    </Metadata>
  </Material>
</MatML_Doc>
```

Fig. 4. A Materials Markup Language example. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

MatML tags have been designed based on entities and attributes in materials science. Users of MatML find it more convenient to store information in this format as opposed to HTML. They find MatML tags more easily interpretable and also find it easier to publish data directly over the Web without requiring conversion from other formats. MatML thus provides a good standard for KR in materials science.

2.5. QuenchML proposal

In spite of the advantages of MatML, it has been observed that this standard is fairly generic. MatML is not enough to capture specific details of processes such as quenching. For example, information related to viscosity (capacity to flow) of a cooling medium used in the process or the geometry of the part being quenched cannot be represented using MatML. This is because MatML has been designed to serve as the XML-based standard for storage of *materials property data* (Begley, 2003). It does not encompass the details of all processes executed with materials. Quenching is a crucial process and generates a huge amount of data that needs to be stored in an integrated format, published worldwide, and analyzed with reference to context, all in a semantics-preserving manner.

This presents the motivation for the development of a domain-specific markup language customized for quenching.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

Thus, a KR standard called QuenchML was proposed at the CHTE at WPI. It was developed with the intention of capturing the semantics of quenching. The initial layout of QuenchML presented at a conference (Varde et al., 2004) was appreciated by the materials science community, serving as the motivation for further enhancement.

QuenchML has been developed such that it can be used in conjunction with MatML. Information from MatML that would overlap with quenching such as quenchant properties and thermal effects can be used in QuenchML. However, QuenchML is not directly integrated with MatML. It is as an independent language, although it indirectly serves to augment MatML with quenching functionality. Note that domain-specific markup languages are typically developed so as to facilitate extendibility by cross-referencing and other means (Yokota et al., 2001; Varde et al., 2010). This applies to MatML as well. Thus, a QuenchML document can easily reference a MatML document and vice versa. This avoids the storage of duplicate information and also facilitates semantics-preserving knowledge exchange.

3. DEVELOPMENT OF QuenchML

3.1. Data modeling for KR

There are real-world entities involved in the quenching process (e.g., the quenchant and the probe). These entities have properties. For example, viscosity is a property of the quenchant. Entities also relate to each other (e.g., the quenchant cools the part). In order to have semantics-preserving KR, it is useful to model these entities and their relationships to illustrate the process and serve as a blueprint for further work.

Several data modeling methods exist in the literature of which entity relationship (ER) diagrams are found to be the most suitable here. An ER diagram serves as a data model to represent real-world entities along with their respective attributes and relationships with each other (Chen, 1976). This is extremely useful in pictorially depicting domain-specific entities in quenching along with their attributes and relationships, thus forming a precursor for designing the structure of the language.

Figure 5 shows a partial snapshot of the ER diagram drawn to model quenching. Here, *Quenchant* depicts the cooling medium used in the process, some of its attributes being *Viscosity* (capacity to flow) and *Composition* (chemical composition of quenchant material). *PartSurface* represents the surface of the part being cooled and has attributes such as *OxideLayer* (thickness of oxide on surface) and *Roughness* (extent to which part is rough). *QuenchConditions* stores details of the apparatus and has attributes such as *Temperature* (part temperature during cooling) and *Orientation* (manner in which part is immersed in quenchant, e.g., vertical, horizontal, diagonal). *Manufacturing* has data about how the part was made. Examples of its attributes are *Welding* (whether part was welded, i.e., fabricated to join materials) and *Stamping* (whether part was stamped during production). *Results* depicts the outputs of the quenching process. Examples of its attributes are *CoolingRate* (speed at which part was cooled) and *Distortion* (extent to which part got distorted while being quenched).

From this ER diagram, we also get an idea of the relationships between the concerned entities. The *Quenchant* cools the *PartSurface*. *QuenchConditions* and *Manufacturing* affect the *PartSurface*. All the entities control the *Results* entity: *Quenchant*, *PartSurface*, *QuenchConditions*, and *Manufacturing*. These relationships as depicted in the ER

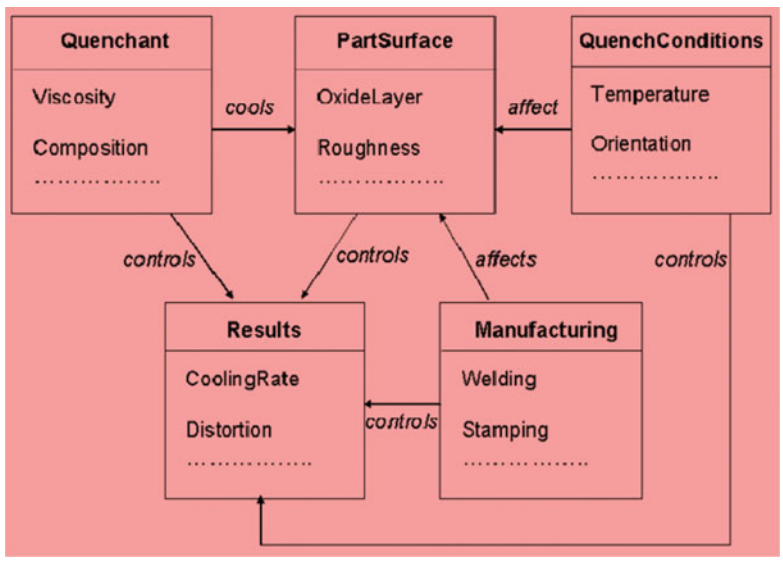

Fig. 5. A partial snapshot of an entity relationship diagram for quenching. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

diagram correspond to the relationships in the quenching process.

3.2. Defining the quenching ontology

Ontology is defined in AI as a formal manner of KR about the world or a part of it through a set of concepts (Russell & Norvig, 2009). Greek roots of the word *ontology* refer to the philosophical study of the nature of existence. An ontological commitment is thus a commitment to an existence claim that is an important aspect of KR. Because capturing knowledge is the key to designing powerful AI systems, it is important to define a good ontology.

There are several issues in ontological design. An important issue is synonymy. Synonyms are two or more words with the same meaning. In quenching, the terms *part*, *probe*, and *workpiece* are synonyms. They all refer to the part being cooled in quenching. We are interested in the surface of the part, the region that gets affected by quenching. Hence, *PartSurface* can also be *ProbeSurface* or *WorkpieceSurface* and can commonly be called *Part*, *Probe*, *Workpiece*, or *Surface*. This similarity is only with reference to context. Another ontological issue is polysemy, or one word with multiple connotations. For example, *Temperature* may refer to the furnace temperature in quench conditions, the part surface temperature while the part is being cooled, or the fluid temperature of the quenchant itself. It is important to clarify this by using the *Temperature* attribute in the appropriate entity and explaining it. Such issues are addressed through ontology.

Figure 6 shows a partial diagrammatic representation of the ontology designed for quenching. This is a high-level subset that relates to some entities identified by the ER diagram shown earlier. We do not show all the entities and attributes. This diagram only helps to convey the purpose of the ontology design. Ontology is formalized using Semantic Web standards Resource Description Framework (RDF) and OWL. RDF is a framework for representing information about resources on the web according to which vocabularies can be defined (World Wide Web Consortium, 2004*b*). OWL is a language useful for describing classes and relations between them inherent in web documents (World Wide Web Consortium, 2004*a*). The use of RDF and OWL along with XML is popular worldwide. These Semantic Web standards are suitable in capturing semantics through ontology definition in markup languages.

For example, synonymy in quenching can be represented using the *sameAs* feature of OWL in conjunction with the *ID* and *resource* features of RDF as shown in the following code snippet:

```
<Quenchant rdf:ID="Quenchant">
     <owl:sameAs rdf:resource="#CoolingMedium"/>
     <owl:sameAs rdf:resource="#Coolant"/>
     <owl:sameAs rdf:resource="#QuenchingMedium"/>
     <owl:sameAs rdf:resource="#Fluid"/>
</Quenchant>
<PartSurface rdf:ID="PartSurface">
     <owl:sameAs rdf:resource="#ProbeSurface"/>
     <owl:sameAs rdf:resource="#WorkpieceSurface"/>
     <owl:sameAs rdf:resource="#Part"/>
     <owl:sameAs rdf:resource="#Probe"/>
     <owl:sameAs rdf:resource="#Workpiece"/>
     <owl:sameAs rdf:resource="#Surface"/>
</PartSurface>
<QuenchConditions rdf:ID="QuenchConditions">
     <owl:sameAs rdf:resource="#InputConditions"/>
     <owl:sameAs rdf:resource="#ExperimentalConditions"/>
     <owl:sameAs rdf:resource="#Conditions"/>
</QuenchConditions>
<Manufacturing rdf:ID="Manufacturing">
     <owl:sameAs rdf:resource="#Production"/>
</Manufacturing>
<Results rdf:ID="Results">
     <owl:sameAs rdf:resource="#Output"/>
</Results>
<Graphs rdf:ID="Graphs">
     <owl:sameAs rdf:resource="#Curves"/>
</Graphs>
```

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

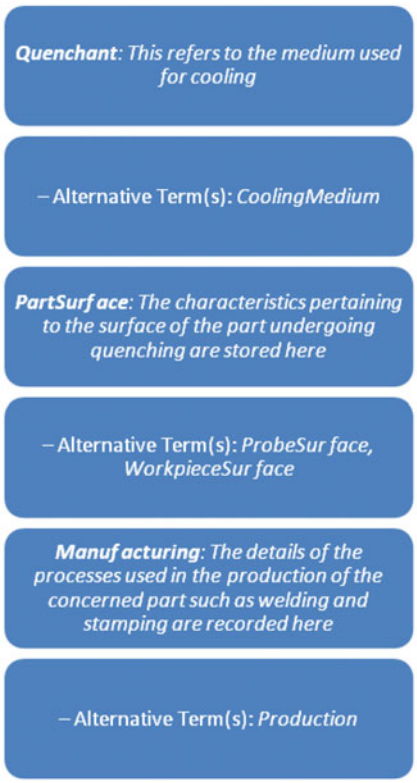

Fig. 6. A partial diagrammatic representation of the quenching ontology. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

Polysemy and related issues can be addressed using the *differentFrom* feature of OWL with the RDF ID from RDF as shown in the examples next. We give one example of the <*Type*> tag used in two different places: within *PartSurface* and *Quenchant*, which refers to the type of material in the part (e.g., steel) and the type of quenchant (e.g., mineral oil), respectively. We give another example of the common word *Curve* appearing in the tags of different curves stored in heat treatment.

```
<Quenching rdf:ID="PartSurface/Type"/>
     <owl:differentFrom rdf:resource="#Quenchant/Type"/>
</Quenching>
<Quenching rdf:ID="#CoolingCurve"/>
     <owl:differentFrom rdf:resource="#CoolingRateCurve"/>
     <owl:differentFrom rdf:resource="#HeatTransferCoefficientCurve"/>
</Quenching>
```

Accordingly, a detailed ontology is defined for quenching. It is not included in this article. However, it can be provided to users upon request.

3.3. Designing the QuenchML schema

The schema provides the structure or grammar for a markup language. Typically in markup language development, after the ontology is formally approved by a team of experts, the first draft of the schema is outlined. Thus, the ER model and ontology serve as the basis for schema design. Each entity in an ER model usually corresponds to one schema element, though there can be additional elements as needed. The ontology helps to include the tags in the schema based on the established system of nomenclature. Schema design is usually subject to revisions.

Figure 7 shows a partial panoramic view of the QuenchML schema that portrays the general structure of the QuenchML language. The tags are nested here such that they depict the relationships between the concerned quenching entities. With reference to data modeling shown earlier, Figure 7 shows that each entity in quenching depicted in the ER diagram more or less corresponds to one element in the schema. The *Results* element in this figure is expanded to show the next level of detail to give an idea of how each element would be structured in the schema. The tags shown here, such as <*HeatTransferCoefficient*> and <*Hardness*>, refer to details of results that users would be interested in observing in the quenching process. For example, the heat-transfer coefficient is a characteristic that refers to the heat extraction capacity of the quenching process. The hardness indicates the extent to which the part has been toughened while being quenched.

Figure 7 is only a part of the entire QuenchML schema. This depicts the manner in which the tags are generally designed in accordance with the entities, attributes, and relationships in quenching to preserve domain semantics. The complete schema can be made available to the concerned users upon request.

4. ENFORCING CONSTRAINTS IN KR

Constraints are defined in AI as consistency conditions that need to be satisfied in order to ensure meaningful search or representation (Winston, 1992; Russell & Norvig, 2009). With reference to our work, a markup language needs to preserve the consistency conditions applicable to restrictions within the domain. For example, a part can be manufactured by either *casting* or *powder metallurgy* but not both (more on this later). Thus, in order to ensure consistency, it is important to preserve this restriction and select one out of these two manufacturing approaches while storing the information.

In order to enforce such restrictions in an XML-based markup language, we use mechanisms in XML that relate to the manner in which a human user would mentally tend

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

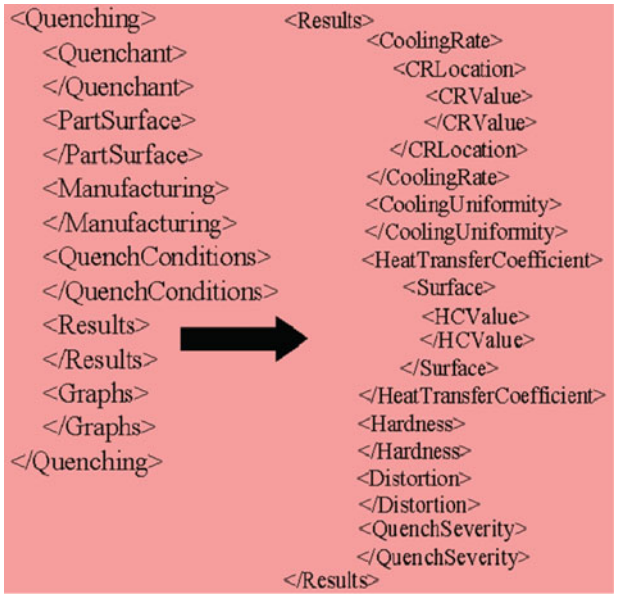

Fig. 7. A partial panoramic view of the Quenching Markup Language schema. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

to enforce restrictions while analyzing data. These are called *XML constraints* (World Wide Web Consortium, 2004*c*). They enable the representation of knowledge adhering to rules such as preserving order and declaring mutual exclusiveness. XML constraints are often used by the Semantic Web community. Some XML constraints have been useful in developing QuenchML and are explained below with suitable examples.

4.1. Sequence constraint for order

The sequence constraint is used for declaring elements to occur in a particular order. It is enforced by enclosing the elements in <*xsd:sequence*> tags (World Wide Web Consortium, 2004*c*). For example, *QuenchConditions* should appear before *Results* because *QuenchConditions* refers to input conditions of the process, and *Results* depicts observations. The conditions obviously impact the observations, so it is important for a user to first read them to understand the impact.

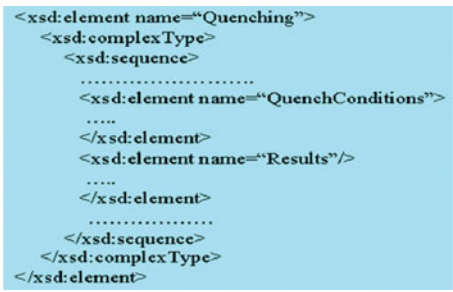

Fig. 8. The use of sequence constraint in the Quenching Markup Language. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

Figure 8 depicts an example of the sequence constraint in QuenchML.

4.2. Choice constraint for mutual exclusiveness

The choice constraint is used to declare mutually exclusive elements (i.e., only one of them can exist). This is enforced by using <*xsd:choice*> in the schema (World Wide Web Consortium, 2004*c*). For example, consider two manufacturing processes: *powder metallurgy* and *casting*.

Powder metallurgy involves physically powdering the concerned material, that is, dividing it into small particles, followed by inserting the powder into a mold or a die for cohesion to produce the part of the desired shape. In casting, a liquid material (not a powder) is poured into the mold or die with the required shape and is then subject to solidification. Thus, it is obvious that a given part can be made by either casting or powder metallurgy but not both (Pellack, 2002).

Manufacturing affects the quenching process, and hence it is important to capture its relevant details while storing quenching data. Thus, the subelements *Casting* and *PowderMetallurgy* in the *Manufacturing* element of QuenchML are mutually exclusive and are enclosed in <*xsd::choice*> tags. Figure 9 is an example of the choice constraint used in QuenchML.

4.3. Key constraint for identification

A key constraint can be considered similar to a key attribute with reference to relational databases and ER modeling (Chen, 1976). This constraint is used to declare an attribute to be an identifier, which means that the attribute must have unique values and cannot have null values.

The key constraint is enforced in the schema by declaring the corresponding attribute as type <*xsd:ID*> and declaring its use as *required* (World Wide Web Consortium, 2004*c*). Figure 10 shows a key constraint in QuenchML.

In this example, *Quenchant* refers to the cooling medium in a quenching process. Its ID is crucial because that serves to uniquely identify the medium (Totten et al., 1993). It is to be noted that an important purpose of conducting quenching experiments is to characterize the quenchants, making the "ID" attribute of the *Quenchant* element even more significant.

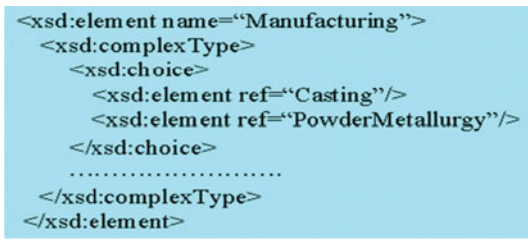

Fig. 9. The use of choice constraint in the Quenching Markup Language. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

```
<xsd:element name="Quenchant">
  <xsd:complexType>
    <xsd:attribute name = "id" type ="xsd:ID" use ="required"/>

    ……..
  </xsd:complexType>
</xsd:element>
```

Fig. 10. The use of key constraint in the Quenching Markup Language. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

4.4. Occurrence constraint for limits

The occurrence constraint is used to limit the minimum and maximum occurrences of an element (World Wide Web Consortium, 2004*c*). It is enforced as *minOccurs=x*, *maxOccurs=y*, where *x* and *y* are the values for the minimum and maximum occurrences, respectively. A *maxOccurs* value of *unbounded* means that there is no upper limit on the number of times the element can occur in the schema. A *minOccurs* value greater than zero means that it is essential to include the element in the schema at least once. Figure 11 illustrates an example of the occurrence constraint utilized in QuenchML.

This example can be interpreted to mean that the *Cooling Rate* element must occur at least eight times, although there is no upper bound on the number of times it can occur. This is because the value of cooling rate must be stored for a minimum of eight points in order to adequately gather the details of the quenching process. However, cooling rate values may be recorded at thousands of points, so there is no upper bound.

For the *Graphs* element, it is not essential for at least one graph to be stored, but it is important to keep the number of graphs to less than three as required in the domain. Typically, two graphs are stored in quenching: a cooling rate curve and a heat transfer coefficient curve. In addition, a cooling curve also may be stored.

These curves are vital in quenching and briefly explained here (Totten et al., 1993). A cooling curve is a plot of temperature (T) versus time (t) during quenching. A cooling rate curve is the derivative of the cooling curve, that is, a plot of cooling rate (dT/dt) versus t. A heat transfer coefficient curve is a plot of heat transfer coefficient (h) versus T, where a heat transfer coefficient characterizes the heat extraction capacity in quenching. It is obtained from a cooling curve by using a heat transfer coefficient equation and is of greatest interest to materials science users. These curves serve as good visual depictions of the quenching process for analysis, and hence it is important to store them.

```
<xsd:element name="Cooling Rate" minOccurs="8"
  maxOccurs="unbounded">
    ……...
</xsd:element>

<xsd:element name="Graphs" minOccurs="0"
  maxOccurs="3">
    ……...
</xsd:element>
```

Fig. 11. The use of occurrence constraint in the Quenching Markup Language. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

5. USEFULNESS OF QuenchML IN KR

5.1. Semantics-preserving storage functionality

QuenchML is proposed to be a worldwide standard for semantics-preserving KR in quenching. It functions as a means to seamlessly integrate various types of data such as numbers and text in a semistructured format.

An XML-based integrated development environment such as XML Spy (Altova, 2010) can be used to provide the editor functionality for data storage within QuenchML. XML Spy can also be used as a plug-in with other environments (e.g., with the Eclipse integrated development environment for programmers who wish to use XML-based tools in conjunction with Java for Web development applications). Other examples of XML editors include XML Marker (Symbol Click, 2001) and Stylus Studio (Component Source, 1996).

Figure 12 is a simple example of how a user would store QuenchML data in a document using XML Spy. This figure shows a part of the whole document (e.g., the dots indicate that more values are recorded for cooling rate). Some more detailed examples can be provided upon request to interested users. As shown in this example, each QuenchML document would be enclosed within a pair of <*QuenchML_Doc*> tags. Several instances of a quenching process can be stored in this document, each instance enclosed within a pair of <*Quenching*> tags. In the given example, we show one instance of a quenching process. The first part of this example as shown in the <*Quenchant*> element can be read as follows: "The quenchant used in the process is called Durixol V35 and is of type mineral oil with viscosity 21.4, its age being 1 year with no degradation." Likewise, the rest of the document can be easily interpreted. Note that user-friendly software can be developed to enable entering all this information through a graphical user interface. The tags would be displayed to the users who could simply type in the contents. This information would then get converted to a QuenchML document as shown above, all this being transparent to the users.

In order to use QuenchML in conjunction with MatML, users can record relevant information using MatML, which serves as the XML for materials property data (Begley, 2003). For example, the property details of the part material being quenched in the given process can be stored in a MatML document. The reference can be made using the element tag called <*Material*> within the MatML document, which would correspond to the subelement tag called <*Material*> within the <*PartSurface*> element in the respective QuenchML

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

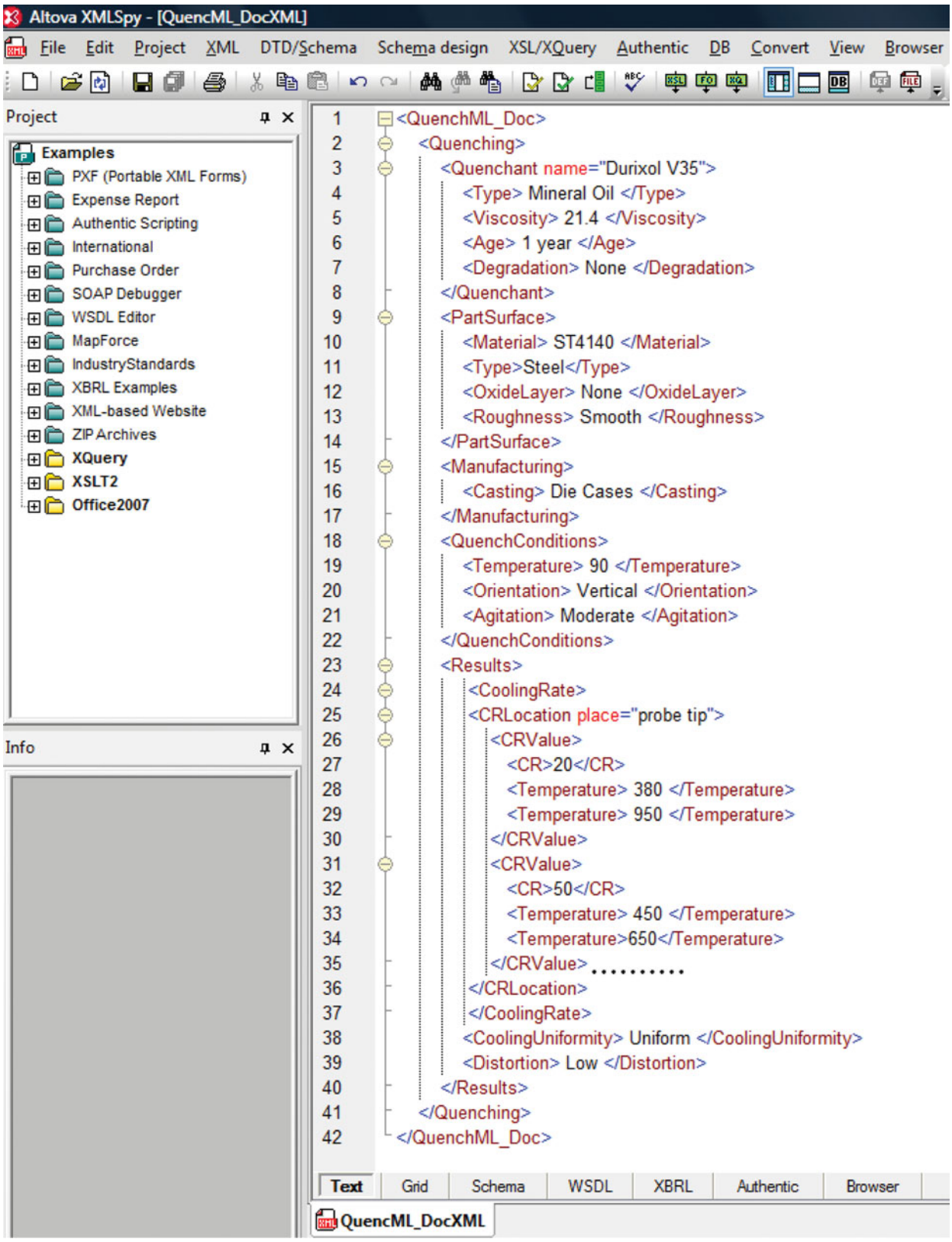

Fig. 12. A partial sample of a Quenching Markup Language document. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

document. This can be further clarified through the ontology, as shown in the following relevant lines of code:

```
<Quenching rdf:ID="PartSurface/Material">
  <owl:sameAsrdf:resource="#MatML/Material"/>
</Quenching>
```

This would avoid duplicate storage and provide easy cross-reference. MatML and QuenchML documents can refer to each other. From this example, it is clear that users can automatically understand the stored data due to the self-descriptive tags and nesting of entities. They would not need much reference to related documentation. In addition, the data is easily publishable worldwide owing to direct storage using an XML-based standard without requiring any conversion, a significant advantage in terms of efficiency. Furthermore, any change would need to be made in only one location on the server used for data storage and would be immediately visible to all users.

5.2. Meaningful IR over the Web

Because QuenchML uses the XML syntax, this serves to provide customized retrieval of information over the Web through XML-based standards. There are many forms of declarative access in the XML family. Of these, we discuss XQuery below with specific examples from QuenchML, giving references to the others.

The XML query language, XQuery, was developed by the World Wide Web Consortium and can be used to retrieve XML data (Boag et al., 2003). Hence, it can be used to query data stored using QuenchML that has been designed with XML tags. XQuery is case sensitive, so it is important to place emphasis on case in QuenchML data.

We demonstrate the use of XQuery through the use of the *for*, *let*, *where*, *order*, *return* (*FLWOR*) expression. FLWOR clauses in XQuery are explained as follows:

- *for* specifies items in the XML document to be selected in the query and is required in the expression,
- *let* is used to create temporary names used in the return and is an optional clause,
- *where* limits the items returned by the query and is also optional,
- *order* is used to change the order of the results and is optional as well, and
- *return* indicates the structure of the data returned and is a mandatory clause.

Based on this, an example of querying QuenchML data directly on the Web using XQuery is shown in Figure 13. This can be entered using an editor such as XML Spy.

The FLWOR expression shown in Figure 13 would traverse the concerned *QuenchML_Doc* and return all the quenchants with viscosity greater than 20.00 ordered by their quenchant types. Such querying is directly possible over the Web because the data has been stored using QuenchML (an XML-based standard) and XQuery works with XML-based formats. If the data had been stored using HTML, then the query would have to be posed over the corresponding database (e.g., in relational format as a SQL query). This would require conversion of the data from HTML to relational formats and vice versa during query processing, a recurrent time-consuming operation to be performed for each individ-

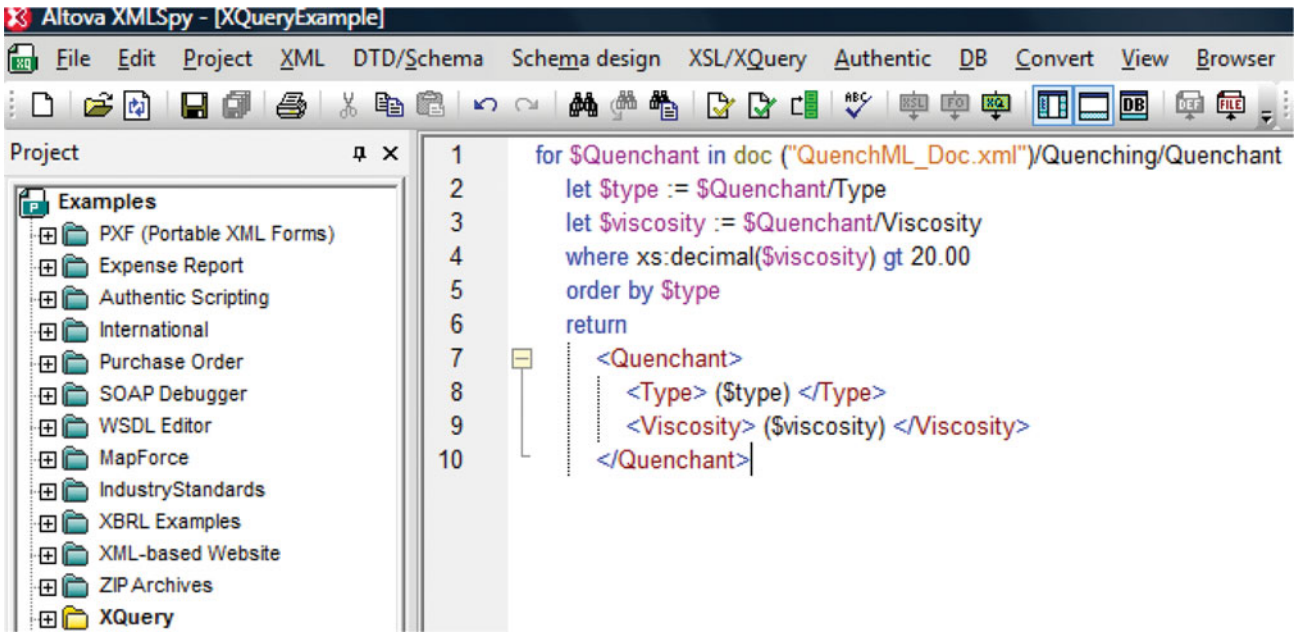

Fig. 13. An example of using XQuery for information retrieval in the Quenching Markup Language. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

ual query. The retrieved information is also more meaningful in QuenchML. Because HTML tags are generic, they do not make the reference to context obvious to the user. In contrast, QuenchML tags provide a quick and easy interpretation by automatically capturing the domain semantics.

Other types of declarative access in the XML family are XSLT: XML Style Sheet Language Transformations (Clark, 1999) and Xpath: XML Path Language (Clark & DeRose, 1999). XSLT and XPath can each be used for IR in QuenchML, although XQuery seems more suitable owing to its similarity with relational querying languages that are already popular. Note that user-friendly tools with interactive graphic user interfaces can be developed using XQuery and others for querying over QuenchML data.

5.3. Advantages of QuenchML

KR in the quenching domain using the semantics-preserving QuenchML language offers several advantages:

1. *Customizing storage:* QuenchML serves to provide a customized form of storage of all the relevant quenching information over the Web for various bodies of targeted users, serving as a nonproprietary Esperanto in the quenching domain.
2. *Avoiding redundancy:* Storage in QuenchML avoids redundancy because QuenchML tags augment MatML without duplication (e.g., data already stored in MatML such as <*BulkDetails*> is not stored again in QuenchML).
3. *Removing ambiguity:* QuenchML helps to present information in a nonambiguous manner by clarifying concepts such as synonyms through a well-defined ontology using Semantic Web standards.
4. *Promoting interpretability:* The use of QuenchML allows better interpretability owing to domain-specific tags that are self-descriptive and are nested to preserve relationships between entities, not requiring reference to related documentation.
5. *Enforcing domain-specific needs:* The design of QuenchML incorporates various requirements specific to the domain through the use of XML constraints, thus incorporating features such as mutually exclusive elements and primary keys.
6. *Facilitating Web IR:* Using QuenchML for data storage in quenching makes it easy and efficient to publish the information over the Web and perform IR using declarative access such as XQuery.
7. *Enhancing knowledge discovery:* Storing data using QuenchML sets the stage for data mining and machine learning, that is, discovering knowledge from sources in quenching and learning from that knowledge to assist future decision making.

Given this, we explain the significance of QuenchML from a data mining and machine learning perspective. QuenchML can help in mining data by preserving semantics, thereby capturing human judgment and giving meaningful results. This topic deserves special attention because it does not seem obvious from the description in the earlier sections. In particular, we discuss how the use of a markup language can help in mining from natural language sources. The explanation we provide for data mining using QuenchML in the quenching domain can also be extended to markup languages in other domains.

6. QuenchML FOR DATA MINING IN QUENCHING

6.1. Data mining and machine learning approaches

Data mining is the nontrivial process of finding novel, useful, and interesting patterns from large data sets to discover knowledge from data (Han & Kamber, 2006). It relates to machine learning that involves development of algorithms to make a machine, more specifically, a computer, learn and evolve behavior usually based on empirical data (Mitchell, 1997). An important goal here is often to support decisions by discovering knowledge and learning from experience. Common techniques for this include association rules, clustering, and classification (Mitchell, 1997; Han & Kamber, 2006). We explain the use of QuenchML in mining quenching data with association rules. Similar arguments can be applied to other techniques (Varde, Suchanek, et al., 2009).

Association rule mining is the discovery of associations of the type $A \Rightarrow B$, where A is the antecedent and B is the consequent. It gained popularity with market basket analysis to identify items commonly purchased together (e.g., *fish* $\Rightarrow$ *chips*). A popular algorithm for association rule mining is Apriori (Agrawal et al., 1993). It uses prior information about frequency items in the data set to discover knowledge about their likelihood of occurring together. Interestingness measures for association rules are rule confidence and rule support. Rule confidence is the probability that the consequent occurs given that the antecedent occurs. Rule support is the probability of the antecedent and consequent occurring in the whole set. Thus, *Rule Confidence=P(B/A)=count (A∧B) / count (A)* and *Rule Support=P(A∧B)=count (A∧B) / count (set).*

Using these measures along with suitable thresholds for minimum confidence and minimum support, the Apriori algorithm mines association rules. These rules can be used to support decisions. For example, in the market basket scenario, we can decide to place fish and chips far from each other so that a customer buying one of these items, being very likely to buy the other, would be tempted to buy more items along the way.

6.2. Mining rules over quenching data

Considering data mining in the context of quenching, if data is stored using QuenchML, it is convenient to use approaches

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

for mining association rules. Consider applying the Apriori algorithm for the discovery of a rule such as $A \Rightarrow B$.

Suppose we have 100 instances of data stored using QuenchML within <*Agitation*> and <*Distortion*> tags. (Agitation is the extent to which the cooling medium is stirred while the part is quenched, whereas distortion is the amount of deformation that occurs in the part.) Consider that 90 of the 100 instances show parts with *high agitation* indicated by the QuenchML data <*Agitation*> *high* </*Agitation*>. Furthermore, 70 of them indicate the occurrence of *high distortion* denoted by <*Distortion*> *high* </*Distortion*>. Among these instances, we find that 60 have both *high agitation* and *high distortion*. On running the Apriori algorithm over this data, we would find an association rule, *Agitation=high* ⇒ *Distortion=high*, with confidence = 60/90 (i.e., 67%) and support = 60/100 (i.e., 60%). This process of association rule mining is summarized in Figure 14. This figure highlights the main aspects of rule discovery at a glance with respect to data storage in QuenchML.

Thus, the use of QuenchML facilitates rule derivation. A big advantage is that the method described above for rule discovery can be applied to plain text sources as well. There are sources of literature in quenching stored in textual formats (i.e., natural language), as opposed to a structured format such as databases. The plain text in such formats can be converted to XML-based formats (i.e., semistructured text) using QuenchML with the help of NLP tools such as the Stanford parser (Klein & Manning, 2003). This semistructured text can then be further processed to a form suitable for rule mining. An algorithm such as Apriori (Agrawal et al., 1993) can then be used to extract rules from these text sources after conversion. The use of semantic tags facilitates capturing relevant information from text sources. This helps to automate knowledge discovery from text that can be useful in AI applications (e.g., expert systems, simulation tools, and intelligent tutors; Jackson, 1999; Russell & Norvig, 2009). QuenchML has been used to develop a system for text mining, which we describe next, in order to exemplify the significance of this markup language.

6.3. The RuleExtractor system for text mining using QuenchML

The importance of QuenchML in data mining and machine learning, especially over text, is evident from a software tool called RuleExtractor (Varde, Aker, et al., 2009) that extracts rules from natural language sources to discover domain knowledge. RuleExtractor performs mining over text sources guided by a markup language. The knowledge discovered is helpful in building predictive and diagnostic systems (Jackson, 1999).

The functioning of RuleExtractor is illustrated in Figure 15, which constitutes its system architecture. RuleExtractor requires that all concerned text sources be integrated into a common repository, serving as a scientific data warehouse (Varde, Aker, et al., 2009). It converts plain text from each source to semistructured text using the Stanford parser for NLP (Klein & Manning, 2003). It does further processing guided by QuenchML tags to convert relevant data in semistructured format to the Attribute Relation File Format (ARFF), required by the data mining tool WEKA (Witten & Frank, 2005). WEKA provides an implementation of association rule mining using Apriori.

In order to emphasize the role that QuenchML plays, we explain the functioning of RuleExtactor. The Stanford Parser used here deploys Treebank tags by the University of Pennsylvania (Marcus et al., 1993). Treebank is a corpus of parsed sentences used to train data-driven parsing algorithms. Tags relevant to our task are simple declarative clause (S), noun phrase (NP), noun (NN), coordinating conjunction (CC), and adjective (JJ). Figure 16 shows a tree structure governing the grammatical structure of the sentence: "Moderate agitation and thick surface implies low distortion tendency." Each node contains a grammatical decomposition of the sentence such that leaves have the words.

The partial output of the parser used by RuleExtractor over this sentence is given below.

```
(S
  (NP
    (NP (JJ Moderate) (NN agitation))
    (CC and)
    (NP (JJ thick) (NN surface) (NN area)))
    (VP
    (VBZ implies)
  (NP (JJ low) (NN distortion) (NN tendency))))
```

From this output it can be observed that subjects or objects like agitation, distortion, and surface are parsed as NN and their descriptors are parsed as JJ tags. It can also be seen that JJ and NN tags are the children of NP.

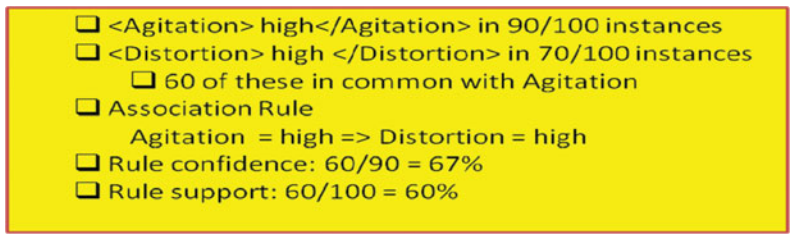

Fig. 14. Association rule mining over Quenching Markup Language data. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

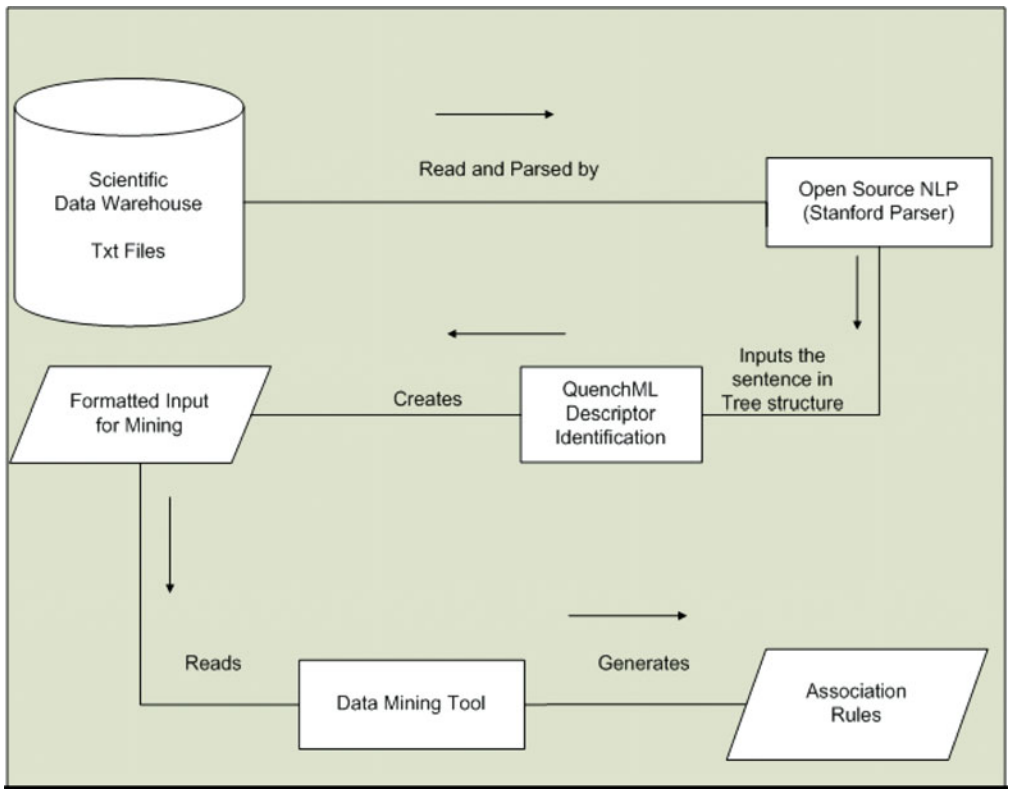

Fig. 15. The RuleExtractor system architecture. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

After conversion to semistructured text, RuleExtractor then checks if there are any predefined QuenchML tags (e.g., <*Agitation*>, <*Surface*>, and <*Distortion*>). The closest matching similar tags are searched (e.g., "surface" in the sentence is similar to the <*PartSurface*> tag in QuenchML). RuleExtractor then determines whether those tags are accompanied by descriptive words. This process continues until the end of the text file is reached. The main process of extraction can thus be summarized by the following steps:

1. Find if NP exists.
2. Find if NN exists under NP.
3. Find if JJ exists corresponding to NP found in step 2.

Finally, after completing these steps, the above plain text gives the following semistructured text.

```
<Distortion> Low </Distortion>
<Agitation> Moderate </Agitation>
<PartSurface> Thick </PartSurface>
```

Here, <*Distortion*>, <*Agitation*>, *and* <*PartSurface*> correspond to the tags in QuenchML, and *low*, *moderate*, and *thick* are the contents of the respective tags. This semistructured text is subject to further preprocessing. It is converted to a format suitable for applying the Apriori algorithm. Because we deploy WEKA, we convert to ARFF. The QuenchML tags are useful here, because these tags form attributes in ARFF and their contents form the data instances. A sample part of our WEKA file thus consists of the following:

@relation Quenching
@attribute Agitation {*moderate, low, high, higher, lower. . . .*}
@attribute Distortion {*low, high. more, less. . . . .*}
@attribute PartSurface {*thick, thin, medium. . . . .*}
@data
moderate, low, thick . . . . . .
low, low, thin . . . . . .

This can be interpreted as follows. We are referring to the "quenching" relation (a relation name is required in ARFF, so we say quenching), such that the attribute "agitation" has possible values "moderate," "low," "high," and so forth. The @data portion of this file contains the actual values that correspond to each instance of the data. Thus, for example, the given instance indicates that agitation is moderate, distortion is low, and part surface is thick. Likewise, we have several instances stored in ARFF that are processed from the corresponding contents of the QuenchML tags after the conversion from plain text to semistructured text.

This ARFF data is then subject to rule derivation using the Apriori algorithm for association rule mining. After running several experiments altering different parameters, we get a set of rules, a sample of which is shown below.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

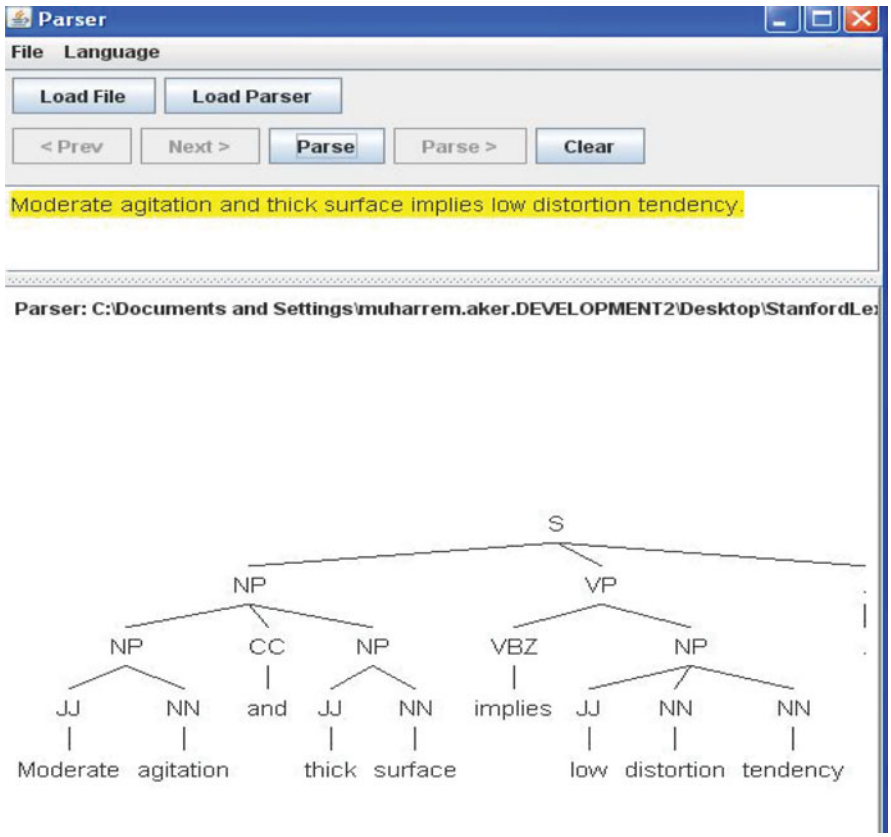

Fig. 16. The tree structure of an example sentence in quenching. [A color version of this figure can be viewed online at http://journals.cambridge.org/aie]

1. *Agitation=high ⇒ HeatTransferCoefficient=high*
2. *Distortion=more ⇒ Agitation=higher*
3. *ResidualStress=yes ⇒ Cracking=likely*
4. *Fixture=improper ⇒ Cracking=likely*
5. *HeatTransferCoefficient=high ⇒ Agitation=high*
6. *Cracking=likely ⇒ Distortion=high*
7. *GrainNature=Nonuniform ⇒ Cracking=likely*
8. *Degradation=no ⇒ Distortion=less*
9. *Viscosity=high ⇒ Distortion=low*
10. *Agitation=higher ⇒ Distortion=more*

These rules are such that they could be used for both predictive and diagnostic analysis in AI systems. In predictive analysis, the goal is to estimate a tendency in advance based on given inputs (Jackson, 1999). For example, the first rule above (i.e., *Agitation=high ⇒ HeatTransferCoefficient=high*) would enable a system to predict that high agitation as an input would lead to a high heat transfer coefficient in the output (i.e., quenching with high agitation would give a high heat extraction capacity). In diagnostic analysis, there is a need to find out what is the cause of a particular occurrence, given a certain output (Jackson, 1999). For example, the second rule shown above (i.e., *Distortion=more ⇒ Agitation=higher*) would help to diagnose that high agitation was one of the causes of high distortion occurring as an output of quenching. The derived rules can therefore be used to develop AI software such as predictive and diagnostic tools.

Domain experts have verified that many meaningful rules were mined by RuleExtractor (Varde, Aker, et al., 2009). This rule automation saves significant time and effort that would be involved if a system developer had to manually infer rules from text sources or by detailed discussion with domain experts. An important factor leading to the success of an automated system like RuleExtractor is that it used semantic tags provided by a markup language such as QuenchML. These tags guided the system as a domain expert would while converting plain text to semistructured text and while selecting the various attributes for mining the data. As a comparative study, we also tried to run experiments without using the tags, and we was found that there were too many words to consider for mining, which did not yield meaningful information (e.g., we got rules such as experiment ⇒ apparatus in the absence of guidance by the markup language). Thus, the markup language certainly provides meaningful interpretation of text for rule derivation.

7. DISCUSSION ON QuenchML USE IN AI SOFTWARE TOOLS

Knowledge discovery over quenching data is useful in building and enhancing AI tools such as expert systems, intelligent

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

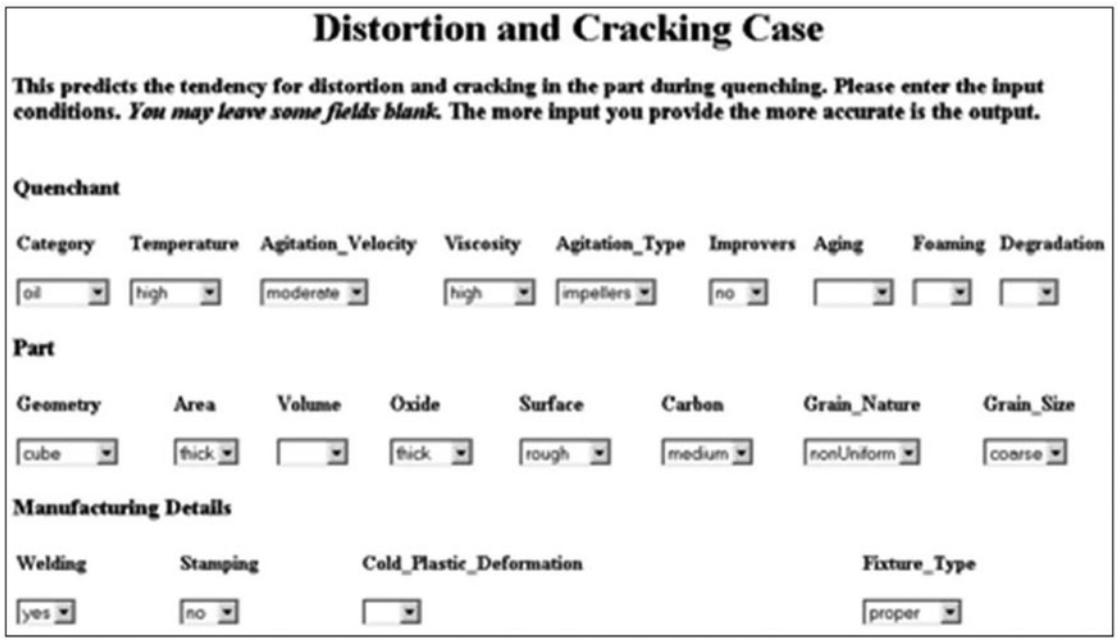

Fig. 17. The input for predicting distortion.

tutors, and fault diagnostic software (Jackson, 1999; Russell & Norvig, 2009). One such tool is an expert system called QuenchMiner (Varde et al., 2003). An expert system is designed to play the role of a human expert in one or more tasks such as providing suggestions, making decisions, predicting answers, diagnosing problems, or performing a narrative role of a storyteller (Jackson, 1999). The QuenchMiner expert system focuses on predictive analysis for decision support. It

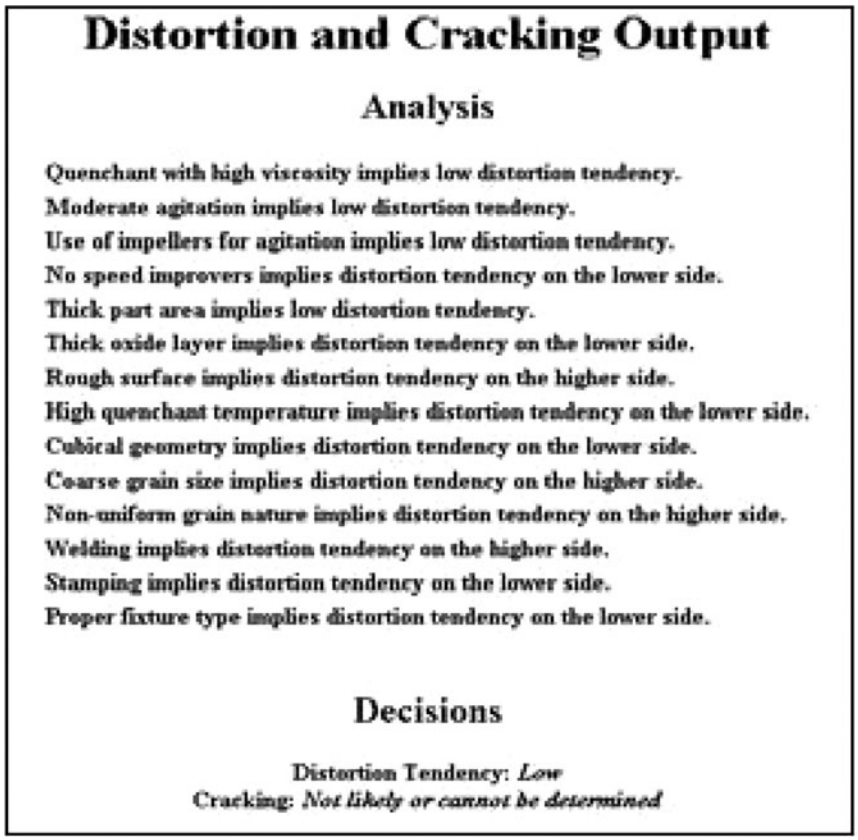

Fig. 18. The output for predicting distortion.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

was earlier developed at the CHTE at WPI. Some of the work in developing QuenchMiner involved considerable manual interaction with domain experts in discovering domain knowledge from text sources of literature. Such knowledge discovery can now be automated using RuleExtractor guided by QuenchML. The expert system can be used for various purposes (e.g., to predict the extent of distortion given several experimental input conditions). Performing such estimation in advance is useful in order to design quenching processes so as to minimize distortion. Quenching conditions can also be chosen accordingly to optimize performance in industrial applications. Figure 17 and Figure 18 present an example of analyzing a distortion case.

The input conditions are shown in Figure 17 and the predicted output in Figure 18. In this example, input conditions in quenching about the quenchant, part, and manufacturing are entered by the user. The QuenchMiner expert system analyzes this information using rules such as *Viscosity=high* ⇒ *Distortion=low*.

QuenchMiner presents the analysis as the predicted output along with an estimate of the extent of distortion and tendency for cracking in this case. Such output can be used to determine whether to use the corresponding quenching conditions while conducting an actual laboratory experiment or while executing detailed quenching processes in the materials science industry.

Other software tools that use QuenchML can be potentially developed for areas such as IR. Various data mining techniques such as clustering and classification (Han & Kamber, 2006) can be used in the development of such tools besides association rules already described here. Without a detailed discussion on these techniques and their targeted applications, we claim that the basic principle would be to use QuenchML for providing the semantic guidance needed to execute data mining and machine learning approaches. Such guidance would otherwise be provided by domain experts, especially in dealing with text sources containing natural language, and consumes considerable time and effort.

Thus, the semantics-preserving nature of QuenchML is useful in domain-specific data mining and in developing AI software tools. The justification presented here for the quenching domain with respect to QuenchML can also be applied to other domains with their respective markup languages.

8. CONCLUSIONS

QuenchML has been proposed to serve as a KR standard in the quenching domain, an important branch of the heat treating of materials. QuenchML follows the syntax of XML and encompasses the semantics of quenching. It seamlessly allows the storage of data and text in an integrated manner using a semistructured XML-based format. It enables better interpretability of data, facilitates IR using XML-based standards such as XQuery, and assists data mining in the quenching domain. QuenchML has been developed closely on the lines of MatML, in order to provide compatibility.

This article describes the development and use of QuenchML. It makes contributions to materials science by proposing a standard to capture domain semantics for storing quenching data, explaining how that standard serves as a communication medium for easy exchange of up-to-date information worldwide, describing how it enables the querying of relevant information in an efficient manner, and also emphasizing its usefulness with respect to knowledge discovery in quenching.

The article contributes to AI by outlining the detailed process of designing a domain-specific markup language for KR, describing semantics-preserving IR using means of declarative access, portraying ontological developments with Semantic Web standards, and explaining constraint preservation using features of XML. It also emphasizes the significance of the markup language in conjunction with machine learning and data mining approaches especially over text sources, and discusses the relevance of this work in AI tools such as expert systems.

ACKNOWLEDGMENTS

The QuenchML effort started at CHTE, part of the Metal Processing Institute in the Materials Science and Manufacturing Program at WPI. The authors of this article are solely responsible for its development. We express our gratitude toward the Quenching Team at WPI for their input. The support and encouragement of the CHTE member companies is also acknowledged. We value the feedback of the Artificial Intelligence Research Group and Database Systems Research Group at WPI. We thank ASM International (the Materials Information Society), the National Institute of Standards and Technology, and the MatML consortium for their work that inspired our research on QuenchML.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases, *ACM SIGMOD*, pp. 207–216.

Altova. (2010). *XML Spy.* Accessed at http://www.altova.com/xml-editor/

Begley, E. (2003). *MatML Version 3.0 Schema* (NIST Technology Report 6939). MD: National Institute of Standards and Technology.

Boag, S., Fernandez, M., Florescu, D., Robie, J., & Simeon, J. (2003). *XQuery 1.0: an XML query language.* World Wide Web Consortium. Accessed at http://www.w3.org/TR/xquery/

Bouvier, D.J. (1995). Versions and standards of HTML. *ACM SIGAPP Applied Computing Review 3(2)*, 9–15.

Carlisle, D., Ion, P., Miner, R., & Poppelier, N. (2001). *Mathematical Markup Language (MathML).* World Wide Web Consortium. Accessed at http://www.w3.org/TR/REC-MathML/

Chen, P.P. (1976). The entity relationship model—toward a unified view of data. *ACM Transactions on Database Systems 1(1)*, 9–36.

Chiu, I., & Shu, L.H. (2007). Biomimetic design through natural language analysis to facilitate cross-domain information retrieval. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21(1)*, 45–59.

Clark, J. (1999). XSL transformations (XSLT) version 1.0. World Wide Web Consortium, W3C draft.

Clark, J., & DeRose, S. (1999). *XML Path Language (XPath) version 1.0.* World Wide Web Consortium. Accessed at http://www.w3.org/TR/xpath/

Component Source. (1996). *Stylus Studio*. Accessed at http://www.componentsource.com/products/stylus-studio

Fahrenholz, S. (2006). *Materials properties thesaurus development.* Paper presented at the ASM International Aeromat Conference, Seattle, WA.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press

Felfering, A., Friedrich, G., Jananch, D., Stumptner, M., & Zanker, M. (2003). Configuration knowledge representations for Semantic Web applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17(1)*, 31–50.
Flynn, P. (2002). *The XML FAQ*. Accessed at http://xml.silmaril.ie/
Guo, J., Araki, K., Tanaka, K., Sato, J., Suzuki, M., Takada, A., Suzuki, T., Nakashima, Y., & Yoshihara, H. (2003). The MML (Medical Markup Language) version 2.3—XML-based standard for medical data exchange/storage. *Journal of Medical Systems 27(4)*, 357–366.
Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques* (2nd ed.). San Francisco, CA: Morgan Kaufmann.
Jackson, P. (1999). *Expert Systems* (3rd ed.). Boston: Addison–Wesley.
Klein, D., & Manning, C. (2003). Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems 15*, 3–10.
Li, Z., & Ramani, K. (2007). Ontology-based design information extraction and retrieval. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21(2)*, 137–154.
Marcus, M.P., Santorini, B., & Marcinkiewicz, M.A. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics 19(2)*, 313–330.
Miller, G.A., Beckwith, R., Fellbaum, C.D., Gross, D., & Miller, K. (1990). WordNet: an online lexical database. *International Journal of Lexicography 3(4)*, 235–244.
Mills, A. (1995). *Heat and Mass Transfer*. New York: Richard Irwin.
Mitchell, T. (1997). *Machine Learning*. New York: McGraw–Hill
Murray-Rust, P. (1997). Chemical Markup Language: a simple introduction to structured documents. *World Wide Web Journal 2(4)*, 135–147.
Pellack, L. (2002). Introduction to materials science. *Issues in Science and Technology Librarianship Spring*. Accessed at http://www.istl.org/02-spring/internet.html
Rilloff, E. (1996). An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence 85*, 101–134.
Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall.
Symbol Click. (2001). *XML Marker*. Accessed at http://symbolclick.com/
Thomere, J., Barker, K., Chaudhri, V., Clark, P., Eriksen, M., Mishra, S., Porter, B., & Rodriguez, A. (2002). A web-based ontology browsing and editing system. *Proc. AAAI 1*, pp. 927–934.
Totten, G., Bates, C., & Clinton, N. (1993). *Handbook of quench technology and quenchants*. Materials Park, OH: ASM International.
Varde, A., Aker, M., & Feldman, A. (2009). *Automated Rule Extraction Over a Scientific Text Data Warehouse Using a Domain-Specific Markup Language* (Technical Report F1209). Montclair, NJ: Montclair State University, Department of Computer Science.
Varde, A., Begley, E., & Fahrenholz, S. (2006). MatML: XML for information exchange with materials property data. *Proc. 4th Int. Workshop on Data Mining Standards, Services and Protocols*, pp. 47–54.
Varde, A., Rundensteiner, E., & Fahrenholz, S. (2010). XML-based markup languages for specific domains. *Web Based Support Systems*, pp. 215–238.
Varde, A., Rundensteiner, E., Mani, M., Maniruzzaman, M., & Sisson, Jr., R. (2004). Augmenting MatML with heat treating semantics. *Proc. ASM Int. Materials Solutions Conf. MatSol, Symp. Developments in Web-Based Material Property Databases*.
Varde, A., Rundensteiner, E., Maniruzzaman, M., & Sisson, Jr., R. (2003). The QuenchMiner expert system for quenching and distortion control. *Proc. ASM International Heat Treating Society Conference, HTS.*
Varde, A., Suchanek, F., Nayak, R., & Senellart, P. (2009). Knowledge discovery over the deep web, semantic web and XML. *Proc. DASFAA*, pp. 784–788.
Winston, P. H. (1992). *Artificial Intelligence*. New York: Pearson Education.
World Wide Web Consortium. (2004*a*). *W3C OWL*. Accessed at http://www.w3.org/TR/owl-guide/
World Wide Web Consortium. (2004*b*). *W3C RDF*. Accessed at http://www.w3.org/TR/rdf-primer/
World Wide Web Consortium. (2004*c*). *W3C XML Schema.* Accessed at http://www.w3.org/TR/xmlschema-0
Witherell, P., Krishnamurty, S., Grosse, I.R., & Wileden, J.C. (2009). Improved knowledge management through first-order logic in engineering design ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 24(2)*, 245–257.
Witten, I., and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). San Francisco, CA: Morgan Kaufmann.
Yokota, K., Kunishima, T., & Liu, B. (2001). Semantic extensions of XML for advanced applications. *IEEE Computer Science Communications 23(6)*, 49–57.

Aparna S. Varde is a computer science faculty member at Montclair State University. She received an MS (1999) and a PhD (2006) in computer science from WPI and a BE (1995) in computer engineering from the University of Bombay. She conducts research in data mining, AI, databases, and science informatics. Dr. Varde has 50 technical publications, 2 software trademarks, 3 research grants, and 14 research advisees (current and former) at all academic levels. Her honors include an associate membership of Sigma Xi for excellence in multidisciplinary research and panel membership in the National Science Foundation in its Division of Intelligent and Information Systems.

Mohammed Maniruzzaman obtained a BS (mechanical engineering) from Bangladesh University of Engineering and Technology in 1987, an MS (mechanical engineering) from Tuskegee University in 1995, and a PhD (materials science and engineering) from WPI in 2000. He worked as an Assistant Professor at Bangladesh University of Engineering and Technology and as a postdoc and later a Research Assistant Professor at WPI, where he performed multidisciplinary research on materials modeling, metal heat treatment, and material characterizations. Currently he is working as a Senior Engineer in the R&D Department at Caterpillar Inc. His research has produced more than 100 technical publications, technical reports, and technical presentations in review meetings and national and international conferences.

Richard D. Sisson, Jr., is currently the Dean of Graduate Studies, the George F. Fuller Professor of mechanical engineering, and the Director of manufacturing and materials engineering at WPI. He received a BS in metallurgical engineering from Virginia Polytechnic Institute in 1969, an MS in metallurgical engineering from Purdue University in 1971, and a PhD in materials science and engineering from Purdue University in 1975. Dr. Sisson's teaching and research focuses on the applications of thermodynamics and kinetics to materials processing and degradation phenomena in metals and ceramics. He has more than 200 publications and more than 200 technical presentations to his credit on these and related topics.

https://doi.org/10.1017/S0890060412000352 Published online by Cambridge University Press