3-2005

An Information Extraction Approach to Reorganizing and Summarizing Specifications

Krishnaprasad Thirunarayan
Wright State University - Main Campus, t.k.prasad@wright.edu

Aaron Berkovich

Dan Z. Sokol

Repository Citation

Thirunarayan, K., Berkovich, A., & Sokol, D. Z. (2005). An Information Extraction Approach to Reorganizing and Summarizing Specifications. *Information and Software Technology, 47* (4), 215-232.
https://corescholar.libraries.wright.edu/knoesis/261

An Information Extraction Approach to Reorganizing and Summarizing Specifications [*]

Krishnaprasad Thirunarayan
Department of Computer Science and Engineering
Wright State University, Dayton, OH-45435.
Email: tkprasad@cs.wright.edu
URL: http://www.cs.wright.edu/~tkprasad

Aaron Berkovich and Dan Sokol
Cohesia Corporation, 8118, Corporate Way, Mason, OH-45040.

August 12, 2004

Abstract

Materials and Process Specifications are complex semi-structured documents containing numeric data, text, and images. This article describes a coarse-grain extraction technique to automatically reorganize and summarize spec content. Specifically, a strategy for semantic-markup, to capture content within a semantic ontology, relevant to semi-automatic extraction, has been developed and experimented with. The working prototypes were built in the context of Cohesia's existing software infrastructure, and use techniques from Information Extraction, XML technology, etc.

Keywords: Knowledge Management, Knowledge Engineering, Information Extraction, Heterogeneous documents.

1 Introduction and Background

Legacy documents in a number of areas of interest are not amenable to automatic manipulation because they were primarily written for human consumption. Furthermore, even if we were to develop authoring techniques that create documents that are simultaneously human sensible and machine processable, it is still unreasonable to reauthor all existing documents. This article deals with coarse-grain information extraction from a collection of legacy heterogeneous documents, and constitutes a case study in Knowledge Management and Engineering, which encompasses techniques and tools for systematically finding,

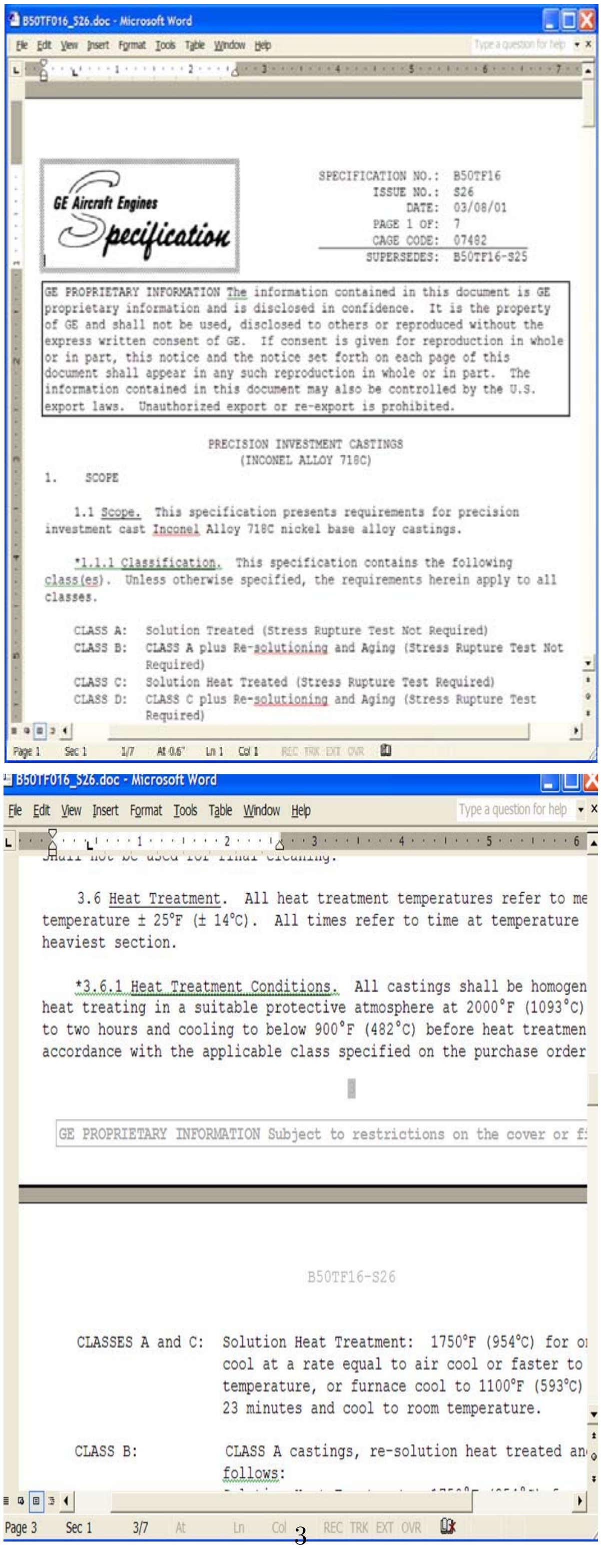
[*] This work was supported in part by NSF SBIR Phases I, II, and IIb Grants DMI-0078525 (1999-2002). The opinions expressed here do not necessarily reflect those of NSF.

selecting, organizing, distilling and presenting information that improves human productivity in a specific domain of interest.

Materials and Process Specifications describe requirements on the processing of a material (alloy) in the mill, and the capabilities that the material should possess eventually. They are critical to materials, aerospace, and automotive industries. Specs basically present numeric data with accompanying text and graphics that explain the quantitative information [45, 46]. They are extensively used by sales personnel, design engineers, manufacturing engineers, and quality assurance personnel. The use of paper-based specs is extremely labor-intensive, quality-impacting, and time-consuming. Even though there is no rigid requirement either on the technical vocabulary used, or on the format, related specs with common origin share similar structure. Specs can be viewed as semi-structured documents with discernible organization and constrained vocabulary. A spec consists of a well-delimited header and a body that is further subdivided into sections and paragraphs. The text in the spec body is not grammatical [49]. Legacy specs are available in hard copy form, and optionally, in electronic form as an MS Word or a PDF document. Figure 1 shows a fragment of a sample spec.

Specs historically have been handled as text documents. However, in order to do any meaningful analysis, comparison, and integration, one should extract numerical, tabular, or graphical data and operate on that data in the fashion intended by its meaning. As part of defining a suitable structure for specs, Cohesia Corporation created the Specification Definition Representation (SDR). SDR is a tree-based declarative language to articulate the semantic view of the components that comprise a spec, and capture the user's interpretation of it. SDR introduced constructs such as *Procedures* to indicate boundaries for standards requirements such as chemical composition, tensile test, melt method, etc. Procedures are composed of elemental *Characteristics* that describe the requirements that are essential for performing the associated process (e.g., carbon content, yield strength, minimum temperature range, etc). In fact, a procedure encapsulates collections of *characteristic-value pairs* glued together using suitable logical connectives. SDR also permits defining a controlled vocabulary of industry terms called the *domain library*, which encodes an approximation to domain ontology and includes lists of names for procedures, characteristics, symbolic values, units of measure, and their inter-relationships. Figure 2 shows the formalization of a sample spec in SDR. The SDR technology has been incorporated into a commercial software system called MASS (Management and Application of Specifications and Standards), and is in use at Fortune 500 companies.

Manual content extraction involves formalizing a paper-based spec in SDR using a domain library. The extractor relies on his/her experience in interpreting a spec, and rendering it in SDR. To be able to understand, query, combine, and manipulate specs in practice, legacy specs should already exist in SDR format, to serve as a foundation to build on. Thus, the cost of data preparation can far outweigh the cost of the manipulation software. So, from business point of view, it is imperative that the workload of a domain expert, that is, human extractor


B50TF016_S26.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
GE Aircraft Engines
Specification

SPECIFICATION NO.: B50TF16
ISSUE NO.: S26
DATE: 03/08/01
PAGE 1 OF: 7
CAGE CODE: 07482
SUPERSEDES: B50TF16-S25

GE PROPRIETARY INFORMATION The information contained in this document is GE proprietary information and is disclosed in confidence. It is the property of GE and shall not be used, disclosed to others or reproduced without the express written consent of GE. If consent is given for reproduction in whole or in part, this notice and the notice set forth on each page of this document shall appear in any such reproduction in whole or in part. The information contained in this document may also be controlled by the U.S. export laws. Unauthorized export or re-export is prohibited.

PRECISION INVESTMENT CASTINGS
(INCONEL ALLOY 718C)

1. SCOPE

1.1 Scope. This specification presents requirements for precision investment cast Inconel Alloy 718C nickel base alloy castings.

*1.1.1 Classification. This specification contains the following class(es). Unless otherwise specified, the requirements herein apply to all classes.

CLASS A:	Solution Treated (Stress Rupture Test Not Required)
CLASS B:	CLASS A plus Re-solutioning and Aging (Stress Rupture Test Not Required)
CLASS C:	Solution Heat Treated (Stress Rupture Test Required)
CLASS D:	CLASS C plus Re-solutioning and Aging (Stress Rupture Test Required)

Page 1 Sec 1 1/7 At 0.6" Ln 1 Col 1 REC TRK EXT OVR
B50TF016_S26.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help

3.6 Heat Treatment. All heat treatment temperatures refer to me
temperature ± 25°F (± 14°C). All times refer to time at temperature
heaviest section.

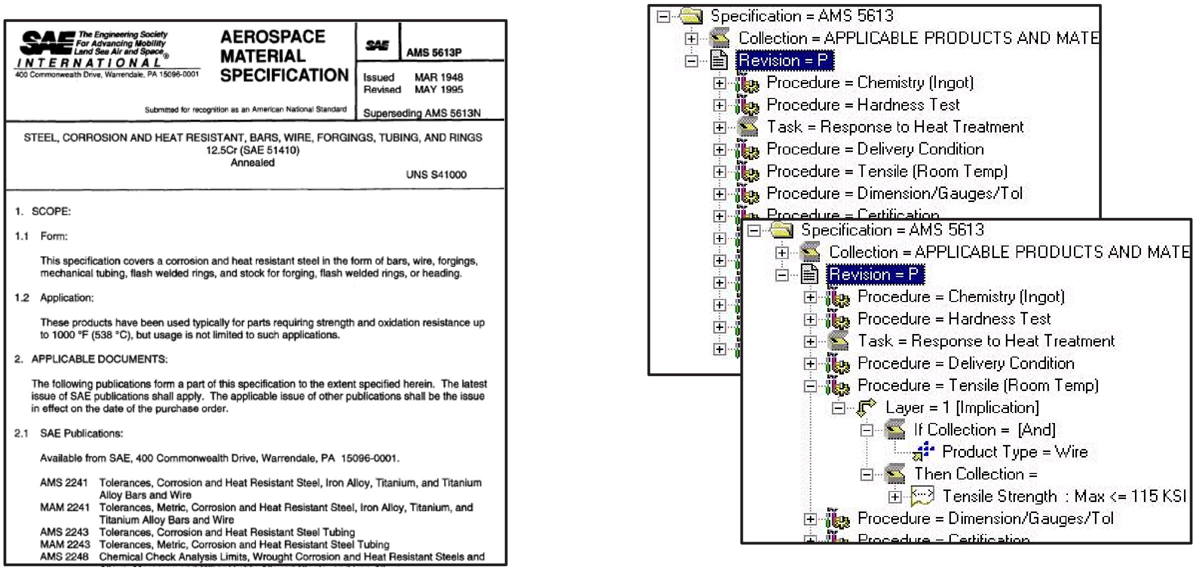
*3.6.1 Heat Treatment Conditions. All castings shall be homogen
heat treating in a suitable protective atmosphere at 2000°F (1093°C)
to two hours and cooling to below 900°F (482°C) before heat treatmen
accordance with the applicable class specified on the purchase order

GE PROPRIETARY INFORMATION Subject to restrictions on the cover or f

B50TF16-S26

CLASSES A and C:	Solution Heat Treatment: 1750°F (954°C) for o cool at a rate equal to air cool or faster to temperature, or furnace cool to 1100°F (593°C) 23 minutes and cool to room temperature.
CLASS B:	CLASS A castings, re-solution heat treated an follows:

Figure 1: A Sample Specification Fragment



SAE INTERNATIONAL The Engineering Society For Advancing Mobility Land Sea Air and Space®
400 Commonwealth Drive, Warrendale, PA 15096-0001

AEROSPACE MATERIAL SPECIFICATION

Submitted for recognition as an American National Standard

SAE AMS 5613P

Issued MAR 1948
Revised MAY 1995

Superseding AMS 5613N

STEEL, CORROSION AND HEAT RESISTANT, BARS, WIRE, FORGINGS, TUBING, AND RINGS
12.5Cr (SAE 51410)
Annealed

UNS S41000

1. SCOPE:

1.1 Form:

This specification covers a corrosion and heat resistant steel in the form of bars, wire, forgings, mechanical tubing, flash welded rings, and stock for forging, flash welded rings, or heading.

1.2 Application:

These products have been used typically for parts requiring strength and oxidation resistance up to 1000 °F (538 °C), but usage is not limited to such applications.

2. APPLICABLE DOCUMENTS:

The following publications form a part of this specification to the extent specified herein. The latest issue of SAE publications shall apply. The applicable issue of other publications shall be the issue in effect on the date of the purchase order.

2.1 SAE Publications:

Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

AMS 2241	Tolerances, Corrosion and Heat Resistant Steel, Iron Alloy, Titanium, and Titanium Alloy Bars and Wire
MAM 2241	Tolerances, Metric, Corrosion and Heat Resistant Steel, Iron Alloy, Titanium, and Titanium Alloy Bars and Wire
AMS 2243	Tolerances, Corrosion and Heat Resistant Steel Tubing
MAM 2243	Tolerances, Metric, Corrosion and Heat Resistant Steel Tubing
AMS 2248	Chemical Check Analysis Limits, Wrought Corrosion and Heat Resistant Steels and

```
Specification = AMS 5613
  Collection = APPLICABLE PRODUCTS AND MATE
  Revision = P
    Procedure = Chemistry (Ingot)
    Procedure = Hardness Test
    Task = Response to Heat Treatment
    Procedure = Delivery Condition
    Procedure = Tensile (Room Temp)
    Procedure = Dimension/Gauges/Tol
    Procedure = Certification
```

```
Specification = AMS 5613
  Collection = APPLICABLE PRODUCTS AND MATE
  Revision = P
    Procedure = Chemistry (Ingot)
    Procedure = Hardness Test
    Task = Response to Heat Treatment
    Procedure = Delivery Condition
    Procedure = Tensile (Room Temp)
      Layer = 1 [Implication]
        If Collection = [And]
          Product Type = Wire
        Then Collection =
          Tensile Strength : Max <= 115 KSI
    Procedure = Dimension/Gauges/Tol
    Procedure = Certification
```

Figure 2: A Sample SDR Extraction

and materials specialist, be reduced to any extent possible, to minimize costs and improve revenue.

Computer assisted content extraction involves semi-automatic recognition of phrases in spec that are associated with requirements, and subsequent synthesis of SDR fragments, to assist an extractor. In general, content extraction can be carried out in two steps: (a) Automatically recognize the structure and the domain library terms present in spec, and (b) Semi-automatically reorganize the spec text to be able to generate SDR from it. The semi-automatic approach can improve the quality and efficiency of extraction by automating some of the routine mechanical tasks.

This article describes our approach and experience with semi-automatic coarse-grain content extraction from specs. The input spec is a heterogeneous document in MS Word format. The domain knowledge and ontology is implicitly given through the domain library. The extraction output is based on SDR but can be in a number of different formats. The heterogeneous document is first converted into a text document preserving the text, the tables, and references to images. XML technology is used for structuring and semantic markup of the text document and for rendering the resulting XML annotated spec into different output formats using XSLT stylesheets [9, 54, 56]. For example, one of the Excel-based output format is suitable for manual post-processing and integration with the spec editor of the downstream product. We have developed a simple content extractor with focus on improving extractor productivity. The prototype incorporated a flexible domain library search engine. The overall approach taken has an industrial rather than academic flavor because we attempted to deal with real-world specs rather than contrived ones. Thus, a realistic yardstick of success is the extent to which mechanical and routine aspects of extraction can be codified and automated, while simultaneously deferring the more difficult and irregular portions to a domain expert.

To summarize the contributions of this article: (1) As far as we know, ours is the first attempt at semi-automatic content extraction from semi-structured, heterogeneous legacy Materials and Process Specifications. In particular, our approach promotes human-in-the-loop for refinement and verification of partial results obtained through automation, by retaining natural correspondence between the spec and its translation, by preserving tables, and by isolating images. (2) We propose and develop a level of extraction that is simultaneously of practical value and amenable to semi-automation. We also sketch the software architecture and engineering details on tools that bridge diversity in authoring organizations. We believe that for domain-specific legacy documents the proposed extraction techniques are a viable starting point. (3) To deal with semantics preserving variations on the limited, domain-specific technical vocabulary, we introduce a new approximate phrase matching algorithm for domain library search that is useful in semi-automatic approach overseen by a domain expert. (4) As a side benefit, the proposed solutions provide case studies in the various uses of XML for document processing such as ease of parsing, as a means to semantic tagging for subsequent machine processing, as a convenient notation for use by domain experts, and as a standard data interchange format.

2 Related Work

The DARPA sponsored TIPSTER program [52] and the Message Understanding Conferences (MUC) [29, 30, 31, 32, 33, 44] provided a major impetus to the progress in Information Extraction technology. The MUC tasks consisted of skimming through free-text articles in a limited domain, and filling in well-specified templates. The template slots and the type or potential values of the slot fillers were defined a priori. Majority of the IE systems submitted for evaluation exploited the English syntax among other features to understand text. In contrast, content extraction from semi-structured specs does not seem to profit much from the English grammatical structure.

The SRI system FASTUS [24], the UMass system [15], and the NYU systems [20, 21, 57] among others made extensive use of patterns in recognizing various tokens and concepts. MUC-7 [33] tasks included development of potentially reusable domain-independent information extraction modules for marking named entities, dates, times, money, and percentages; determining co-references; and marking relationships among different elements such as employee_of, product_of, location_of, etc. In fact, GATE [19] provides Java implementations of some of these modules in an integrated development environment. The activities pursued in the context of content extraction from specs are analogous in spirit to the above activities. However, content extraction from specs requires recognizing domain library terms or its equivalent, and cannot directly benefit from the NLP information extraction tools available in GATE.

Information extraction consists of defining the form of extraction rules, acquiring them by hand-crafting or by machine learning, and then applying them. In terms of the design space for extraction and transformation from semi-structured documents studied in Crespo et al [7], materials and process specs have the following coordinates: they have *implicit structure*, and are *single documents with references*; they are *partly irregular* and *relatively stable*, but *can contain format and content errors*. The work reported here attempts semi-automatic content extraction from a spec by exploiting domain specific background information in the domain library to elicit the semantics, and then transforming the spec as desired.

Typically, extraction rules formalize linguistic features of the desired information, or the context surrounding the desired information (for example, the enclosing HTML tags). Extraction rules can be generated using machine learning algorithms [34]. Automatic Dictionary Construction tools, such as Autoslog [43], CRYSTAL [48], etc work on free-text [27], while systems such as RAPIER [5], SRV [16], WHISK [49], and Wrapper Induction Systems such as STALKER [35], RoadRunner [6], Lixto [2], etc work from online HTML documents. Ashish [1] develops mediators for extracting and integrating information from distributed sources. In the realm of specs, the form of extraction rules and their automatic learning is still an open problem of great practical interest and value. Furthermore, there are no pre-existing corpus of annotated specs to learn from.

In order to better appreciate the relationship between our work and the works of others, we recapitulate the taxonomy for characterizing the Data Extraction Tools proposed by Laender et al [26]. In this taxonomy, the same tool can belong to multiple groups.

- *Languages for Wrapper Development*: These are specialized languages that assist in wrapper construction. We use a simple XML-based language for describing the format of various families of specs for extraction.

- *HTML-aware Tools*: These tools rely on inherent structural features of HTML documents for synthesizing extraction rules and performing data extraction. Specs are not HTML documents.

- *NLP-based Tools*: These tools apply parts-of-speech tagging and lexical semantic tagging to assist in deriving extraction rules. Specs domain does not benefit much from the Natural Language processing techniques.

- *Wrapper Induction Tools*: These tools generate delimiter-based extraction rules derived from training set. These tools do not rely on linguistic constraints, but rather on formatting features that implicitly delimit the desired information. Our approach is based on extracting content from its explicit presence or implicitly via domain ontology as opposed to infer it from the formatting features in the surrounding context.

- *Modeling-based Tools*: Given a target structure for the desired information, these tools try to locate in Web pages, portions of data that implicitly conform to that structure. These tools are particularly suited for documents that have clearly identifiable inherent structure (for example, bibliographic references) [42]. The overall layout of a spec can be formalized but the content of a spec does not exhibit structural regularity that can be exploited for extraction.

- *Ontology-based Tools*: In contrast with the above tools that rely on the presentation features of the data to generate extraction rules, the ontology-based tools use domain ontology to locate data and assimilate information. Our tool falls into this group. Even though specs can be regarded as data rich and narrow in ontological breadth [10], fine-grained content extraction is still very complex. Note also that the domain library, which realizes the ontology, has been constructed and evolved over time using phrases in legacy specs.

Information Extraction technology has also resulted in commercial products and search engines that classify and search documents based on the subject matter. Unfortunately, these products cannot be used directly because there are significant differences between processing general topic documents and highly technical specs, and between classification and reorganization or summarization.

Natural Language Processing (NLP) Technology has been successfully applied for the maintenance of textual databases containing manufacturing plans and procedures at Boeing [36, 37]. Specifically, for tractability reasons, the syntax and semantics of the text was controlled prior to populating the databases by using an authoring tool. NLP tools were used to effect mass changes in textual databases for the circumscribed domain of chemical treatment, prime and finish operations, to correct errors and update information. The work reported here resembles that described in Obrst et al [36, 37] in so far as they attempt to exploit domain specific background information to elicit the semantics, and use language processing tools to transform documents as desired.

The spirit of our work is similar to what is described in Fensel et al [11, 12, 23]: In order to apply AI in realisitic, large scale document processing applications, it is necessary to make explicit machine-processable semantics of sources. In fact, it was heartening to discover that the techniques used and the infrastructure built by Cohesia over the last decade, motivated by the perceived needs of B2B and B2C e-commerce in Materials-related industry, resonates well with the issues cropping up in the Semantic Web context [14]. The research done in the context of *OntoWeb*, a European Union founded project about Ontology-based information exchange for knowledge management and electronic commerce [38], is particularly relevant. In fact, we believe that such lines of research are important for the long term success of the Semantic Web initiative [3, 51, 14] and interoperability issues in the Web Services context [13, 8]. And the work reported here is just the first step in that direction.

3 General Approach to Content Extraction

The input spec is heterogeneous, consisting of text, tables, and images in MS Word format. The domain library, which approximates domain ontology, consists of technical phrases condensed from legacy specs. The extraction consists of recognizing a subset of domain library terms and their relationships in the spec, and then associating them with applicable fragments of the spec. The eventual output of extraction can be in plain text, in XML, in SDR, or in Excel form, depending on the application.

In order to programmatically manipulate a spec, it is convenient to translate it into plain text first. To ensure direct correspondence between the original spec and the translated text, it is also important to retain the context of an image and preserve the grid layout of a table.

A spec consists of spec header and spec body. The different authoring organizations create specs that differ in the location of the header information and its surrounding context. However, for each authoring organization, the header information context is more or less fixed. To extract header information, extraction rules formalizing the context of the header information in terms of the start and end markers can be used with a driver program. These extraction rules should be simple enough to be easily created and changed by a domain expert. Each organization can have different domain library for interpreting spec body. To extract content, corresponding spec phrase is recognized and mapped to equivalent domain library term. The words to be mapped to a domain library term may be neither contiguous nor appear in the same order. The words may even be shared among multiple domain library terms. Thus, the string matching algorithm to be used for recognizing phrases that map to domain library terms needs to be robust with respect to afore-mentioned eventualities. Recall that domain knowledge is available only in terms of the domain library, and there are no pre-annotated specs available for machine learning.

The extraction output consists of fragments of spec text conditioned on boolean expressions over domain library terms. The form of output can be in plain text, in XML, in SDR, or in Excel form. Due to the complex nature of the specs, the final extraction is usually inspected by a domain expert in a Spec Editor for correcting any errors.

4 Content Extraction from Specifications

Here we explore techniques underlying a tool for coarse-grain content extraction from domain-specific heterogeneous documents (materials and process specs). In particular, the approach to extraction depends on two orthogonal issues pertaining to documents: the content and the form.

- Usually, a spec defines requirements for several alloys, and the requirements can further depend on a number of parameters such as product type, product dimension, cross-section, spec class, etc. Thus, a spec can be viewed as a compact description of a collection of *primitive* specs conditioned on suitable criteria. Coarse-grain extraction involves reorganizing spec content, to enable determination of primitive specs, in a form usable with Cohesia's existing software infrastructure. Domain knowledge has been codified as a domain library of terms, compiled and evolved over a decade from commonly occurring technical phrases in concrete specs.

- Recall that a spec is a heterogeneous document consisting of text, numeric data, images, and tables. For extraction purposes, it is important to delimit different forms of data and develop appropriate techniques to handle each of them. A spec available in electronic form as an MS Word document (as opposed to a PDF document) is better amenable to semi-automatic handling.

In what follows, we discuss techniques and tools to abstract and map:

- a heterogeneous spec in MS Word format into a text form,
- a document phrase into domain library terms, and
- finally, the spec text into a sequence of conditioned text.

4.1 Separating Tables and Images from the Text

In order to focus on the semantic content of a spec, it is necessary to remove presentation details from the MS Word document. The simplistic approach of saving MS Word document as a text document loses the embedded images and trashes the tables. Saving it as an HTML document contains too many superfluous tags. To enable automatic processing of text while deferring tables and images for manual handling, it seems reasonable to delimit text from images and tables, abstract tables as rectangular grids of numbers and creating links to embedded image files. The following steps can be taken to obtain a spec in ASCII plain text form:

1. Identify paragraphs properly and replace non-ASCII characters in MS-Word document with their ASCII encodings by running appropriate VB macros.
2. Save the MS-Word document in RTF format.
3. Convert the RTF document into an XML document using IBM developerWork's Majix (a Java application) that replaces an image with image tags and a reference to an image file, and preserves the structure of the table using table tags.
4. Finally generate a plain text document by removing image tags, and replacing table tags by appropriate indentation.

Specifically, MS-Word to ASCII converters were created:

- to properly interpret paragraph breaking points and include additional line separators where needed,
- to encode non-ASCII characters such as ^{o}, ±, etc into ASCII,
- to preserve context for the embedded images, and
- to properly format tables by aligning columns with column breaks in ASCII representation, for readability, and
- to generate spec in plain text form that could then be used by the extraction utility.

4.2 Domain Library Search Engine

The domain library is intended to approximate domain ontology and has been constructed using technical vocabulary of the current and legacy materials and process specifications. Domain library terms, abbreviated as DLTs, are classified into categories such as procedures, characteristics, controlled terms, etc based on their role, with each term permitted to have multiple roles. Domain library also maintains relationships among various DLTs. Examples of common relationships are — characteristics appearing within certain procedure, values associated with certain characteristic, broader and narrower terms, aliases, etc.

In manual extraction, an extractor maps a phrase associated with a material requirement in a spec to a domain library term by guessing the corresponding term and verifying it against the domain library. For instance, a phrase in a spec may not be identical to its name in the domain library, as illustrated by the recurring patterns and variations given below:

- Chemistry, Composition, Chemical Composition, Chemical Analysis, ...
- Rejection Criteria, Criteria for Rejection, Causes for Rejection, Not Acceptable, ...
- Certification, Report, ...
- Bend Test, Bending, ...
- Delivery Condition, Process/Surface Finish, Temper, ‘as received by purchaser’, ...
- the height of a box, box height, ...
- Qty = Quantity
- Temp = Temperature
- Examination = Exam

A typical domain library[1] has about 10,000 terms. The effectiveness of the search, and thereby the productivity of the extractor, can be substantially improved by accepting an arbitrary phrase and automatically determining the relevant DLT(s) contained in it. The overall goal is to develop an approximate matching algorithm to recognize equivalent technical phrases that map to the same DLT, while balancing recall and precision [28, 41].

Initially, we considered mapping a well-delimited spec phrase to an equivalent DLT, by defining a suitable normal form for phrases and then searching for the normalized phrase in the normalized domain library. Normalization was achieved by using the following sequence of operations with two phrases being equivalent if their normal forms match exactly:

1. Delete superfluous words and change words to lower case.

2. Expand acronyms and abbreviations.

3. Replace words by the synonymous preferred forms.

4. Stem each word and then sort the entire list of words alphabetically.

The conversion to normal form is efficient, but is not satisfactory in general, when we experimented with popular stemming algorithms available such as due to Porter, Lovins, etc, a conclusion shared by several other researchers [25, 40]. We also discovered that prefixes (e.g., electro-, spectro-, re-, etc) needed to be separated to recognize the root. In general, technical terms are often compound words, and new terms are created by gluing existing term fragments [18, 39].

Our new heuristic algorithm has been designed with the following observations in mind:

1. An input phrase may contain multiple DLTs.

2. The DLT words contained in the input phrase may not appear contiguous.

3. Consonants are significant, and correct spellings may differ in the vowels.

4. Robustness with respect to misspellings such as transposition of letters or missing vowels is desirable.

5. Ordinary stemmers do not work for technical words appearing in the DLTs. Instead, customized tables of prefixes and suffixes were developed to deal with situations that arise in practice. Furthermore, normalization is dynamic, and is done by pitting pairs of phrases against each other, and then abstracting their differences.

6. Pragmatic considerations encouraged us to make the algorithm skewed towards recall rather than precision as a domain expert is assumed to be overseeing the outcome.

[1] Cohesia creates and maintains domain libraries for in-house use and for use by its clients such as GE, Alcoa, Allvac, etc.

4.2.1 Algorithmic Details: Expression Matcher

A customized string matching algorithm for limited vocabulary domain was developed to map an arbitrary phrase/sentence to a list of domain library terms (DLTs) contained in it, embodying the observations given above. The empirically derived mapping tries to determine approximately semantically equivalent DLTs.

The main steps of the heuristic algorithm for DLT matching are sketched below. (Note that the inputs are assumed to have the abbreviations expanded and aliases normalized to preferred term).

```
List[Phrase] dl, dlwm, inwm;
Phrase ip;     Integer mt;
List[Phrase] dlts;

Step 1: Read list of Domain Library Terms (phrases) into dl.
Step 2: Tokenize and construct list of (non-stop) words in the
        domain library terms into dlwm, maintaining links from
        each word back to the DLTs it appears in.
Step 3: Read-in input phrase and match-threshold into ip and mt,
        respectively.
Step 4: Tokenize and construct list of non-stop words into inwm.
Step 5: Determine matching words in dlwn corresponding to each word
        in inwm. (This step is discussed in detail below.)
 Step 6: Construct list of DLTs that contain these matching words.
        Evaluate each DLT and filter good approximations included in ip.
```

Words are matched at four levels: case-insensitive, normalized (based on consonants), sorted normalized (based on sorted string of consonants), and root-oriented. The exact match requires the case and whitespaces to match, while case-insensitive match converts the inputs to lowercase prior to matching. Such matches are useful when searching for names of the chemicals or materials verbatim. Normalized match deletes all vowels and special characters (such as '-', '?', etc) prior to matching. Sorted normalized match orders the consonant string before matching. Such matches add robustness and improve productivity without jeopardizing soundness in the context of limited vocabulary search overseen by a domain expert.

If these checks do not yield a match, the strings are then tested for common roots by considering variations due to affixes. For this purpose, sorted normalized representations are compared and common letters are eliminated for each string. For example, the words 'certify' and 'Certification' yield 'crtfy' and 'crtfctn' upon normalization, which can then be sorted to yield 'cfrty' and 'ccfnrtt'. Eliminating common letters results in 'y' and 'cnt'. Of these remaining letters, if they constitute differences accountable using the following customized table of viable suffixes and prefixes, then a match is declared.

- `Remainder_Suffix` table lists the remaining letters and expected suffix.

- Remainder_Prefix table lists the remaining letters and expected prefix.
- Remainder_Prefix_Suffix table lists the remaining letters, expected prefix and expected suffix.

Specifically, a match is declared if the remaining letters of both the words are present in one of these tables, and the word has the expected prefix and/or suffix given in the table. For example, the words ‘certify’ and ‘Certification’ match because both (‘y’, ‘y’) and (‘cnt’, ‘cation’) appear in Remainder_Suffix. Fragments of the three tables incorporated into the implementation are shown below, which presents a workable solution around the limitations of the stemming algorithms for the Materials and Process Specs application context:

```
Remainder_Suffix[] = {
"d", "d",
"s", "s",
"mnt", "ment",
"mnst", "ments",
"nt", "tion",
"nst", "tions",
"cnt", "cation",
"cnt", "cations",
"cnst", "cations",
"y", "y",
"ct", "cate",
"ct", "cates",
"ct", "cation",
"cst", "cates",
"cst", "cations",
"n", "ion",
"n", "ions",
"ns", "ions",
"c", "ance",
"g", "ing",
"cl", "cal",
"rsty", "stry",
"ly", "ly",
"rst", "stry",
"cll", "cally",
"sy", "stry",
"rty", "stry",
"cl", "cals",
"gny", "ying",
"gn", "ing",
"gy", "ying",
};

Remainder_Prefix[] = {
"cprst", "spectro",
"clrt", "electro",
"ps", "spectro",
"l", "electro",
"cprt", "spectro",
"crt", "electro",
"r", "re",
"lw", "low",
"n", "un",
};

Remainder_Prefix_Suffix[] = {
"cclp", "spectro",
"cclp", "cal",
"ccll", "electro",
"ccll", "cal",
"rs", "re",
"rs", "s",
"dr", "re",
"dr", "ed",
"dn", "un",
"dn", "ed",
"gnr", "re",
"gnr", "ing",
"glnw", "low",
"glnw", "ing",
"gr", "re",
"gr", "ing",
"glw", "low",
"glw", "ing",
};
```

The following abstract code makes the tests more explicit.

```
Step 0:   Let w1 and w2 be the two words.
Step 1:   sw1 = Sorted_Normalized(w1);  sw2 = Sorted_Normalized(w2);
Step 2:   r1 = Letters_Difference(sw1,sw2); r2 = Letters_Difference(sw2,sw1);
Step 3:   Match(w1,w2) =
            ( IsSuffix(Remainder_Suffix(r1),w1) or
              IsPrefix(Remainder_Prefix(r1),w1) or
              HasPrefixSuffix(Remainder_Prefix_Suffix(r1),w1) )
            and
            ( IsSuffix(Remainder_Suffix(r2),w2) or
              IsPrefix(Remainder_Prefix(r2),w2) or
              HasPrefixSuffix(Remainder_Prefix_Suffix(r2),w2) );
```

Examples of sets of matching words include {`certification`, `certificate`, `certify`, `certified`, `certifies`, `certify`}, {`require`, `required`, `requirement`}, etc.

Let a pair of words match to one of four different degrees, corresponding to case-insensitive, normalized, sorted normalized, and root-oriented match.

(These are actually empirically determined weights between 0 and 1.) Then, the degree of match of an input phrase with another phrase is obtained by averaging the maximum degree of match between each non-stop word in the input phrase with non-stop words in the other phrase. In the context of domain library searches, an input phrase is efficiently matched with DLTs (phrases), yielding a list of matching DLTs sorted on the basis of degree of match. Ideally, one would expect the DLT with the highest degree of match to be the desired interpretation. In practice, it is desirable to have a domain expert verify it. It is unrealistic to expect the search to be robust in the face of tricky constructions, arbitrary semantics preserving rewrites, or non-trivial context-sensitive interpretations. In future, we will be working on techniques to refine degree of match, to improve precision.

4.3 Coarse-grain Extraction

The text of a spec can be viewed and annotated to different levels of detail (granularity) in order to make explicit mechanically processable information. The text is human sensible, while the annotation, which summarizes and abstracts text, is machine comprehensible. The different extraction granularity reflects the different ontologies[2] used in the industry and in companies such as GE, Alcoa, Rolls Royce, Pratt and Whitney, etc. The feasibility and the difficulties encountered in automating an extraction task depend on its granularity. Ironically, fine-grained extractions are neither necessary nor desirable for all applications.

A typical materials and process spec contains requirements for making and testing a variety of alloys. A typical customer order specifies the desired material in terms of the specs it must conform to, and a collection of domain-specific product parameters such as product type, spec class, product dimension and cross-section, etc. A fundamental operation of interest on a spec is the determination of applicable fragments of the spec for a customer order. The goal of coarse-grain extraction is to convert a spec into a form that can be evaluated against the order parameters to determine applicable fragments. So we explore techniques for generating conditions for the applicability of a spec text fragment.

To balance the commercial viability of the extraction task and its tractability, the following extractions were explored:

Basic Extraction: This involves the identification of header information such as spec name, spec title, organization name, revision information including revision date, etc, and filtering the spec text, to be subjected to further scrutiny later.

Level 1 Extraction: A spec is transformed into a possibly, nested sequence of conditioned notes of the form `If CONDITIONS Then [Note = " ... "]`,

[2] According to Gruber [22], an ontology is an explicit specification of a conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purpose.

where the note contains contiguous block of spec text. These extractions are cheap to produce because the detailed requirements are still in text. They can be used to reorganize/filter a spec, but they have insufficient structure to allow machine manipulation and detection of conflicting requirements. In practice, such extractions are used with MASS FAI (First Article Inspection).

Level 2 Extraction: Procedures group related requirements. A procedure can appear in a spec explicitly or implicitly through related characteristics. In Level 2 extraction, a spec is transformed into a sequence of procedures, with the spec text relevant to each procedure, structured as a sequence of notes, serving as the procedure body. These extractions are finer grain than Level 1 but has insufficient structure to allow numeric result reporting or machine manipulation and detection of conflicting requirements. Level 2 Extractions are further refined by a domain expert before being used with MASS Order Product.

. . .

Level * Extraction: The requirements expressed in numeric and symbolic terms are captured formally in a machine processable form in SDR to enable automatic analysis, combination and conflict resolution of requirements. At this stage of evolution and maturity, Level * Extraction cannot be automated, but must be carried out manually by domain experts.

To ensure commercial viability and tractability of automating the extraction tasks, only Basic Extraction and Level 1 Extraction were implemented. (Level 2 Extraction was also explored, but we do not discuss the details any further in this article.) Basic Extraction can be carried out by developing extraction rules for determining header information by formalizing the context surrounding the desired information in terms of the start and end delimiter keywords. These rules are developed for each spec family and for each spec authoring organization.

In what follows, we explain Level 1 Extraction in more detail. Specifically, we abstract the intrinsic nature of translation that makes it amenable to semi-automation in Section 4.3.1. We discuss heuristics used for generating conditions by focusing on the document structure and by creating suitable scope rules for applicability of conditions in Section 4.3.2. Finally, we sketch the implementation details for Level 1 Extraction in Section 4.4 [47].

4.3.1 Literal Translation

Conceptually, every piece of information in SDR formalization of Level 1 Extraction owes its existence to a phrase in spec and possibly in domain library. For ease of verification, it is important to maintain a one-to-one correspondence between fragments of spec and its translation. Whenever feasible, such translations are said to be *literal*. The existence of a literal (linear) translation depends on the target language and the translation scheme. To see this dependence, consider the sources of non-linearity:

1. The extractions can contain duplicated information that are shared in spec.
2. The extractions can rearrange information present in spec.
3. Tables and footnotes abbreviate information in irregular and complex ways.

Furthermore, a semi-automatic approach to extraction is feasible only if the automatically generated partial translations are intelligible to the domain expert in the context of the spec, for subsequent modification. A semi-automatic approach emphasizes the need for literal extraction.

4.3.2 Level 1 Extraction

Related specs from an authoring organization share similar structure. Normally, a spec document consists of a sequence of sections. A section is associated with a section heading including section number and a section body in the form of paragraphs, possibly containing nested subsections. A paragraph is a piece of text delimited by a pair of blank lines. These do not have any heading associated with them. In general, each (sub)-section or paragraph can contain requirements applicable to several different final products, and explicit references appear in situations where the applicability is limited to a few products.

In order to formalize conditioned notes, we need to propose a structure for the conditional expression and the note it qualifies. The conditional expression can be formed using characteristic names, constants, relational symbols (e.g., '=', '>', etc), and boolean connectives (e.g., 'and', 'or', etc) in the standard way. The note can be defined in quanta of (sub)-sections and paragraphs. Thus, there are two important technical problems to be solved for carrying out Level 1 Extraction: (1) Identification of the values of a characteristic that can appear as a condition, and (2) Transformation of the relevant spec text into a sequence of conditioned notes.

In what follows we explain and illustrate the kinds of problems tackled.

Section Numbering: Recognition of section numbers, separate from numeric data, is complicated by the fact that sections can be nested. For example, if the current section number is '3.10.2', the next section numbers can be '3.10.2.1' if it is a *deeper* section, '3.10.3' if it is *at the same level*, and '3.11' or '4' if it is a *shallower* section. The `nextSectionNumber`-relationship among pairs of dot separated sequence of numbers (which conforms to regular expression $\{1,2,\cdots,9\}\{0,1,\cdots,9\}^*(\bullet\{1,2,\cdots,9\}\{0,1,\cdots,9\}^*)^*$) can be formalized as follows [50]. (For readability reasons, the ellipses have been retained.)

$$\forall i > 0 : \forall n_1 > 0, \ldots, \forall n_i > 0 :$$

$$\texttt{nextSectionNumber}(n_{1\bullet}n_{2\bullet}\cdots{}_\bullet n_i,\;\; n_{1\bullet}n_{2\bullet}\cdots{}_\bullet n_{i\bullet}\mathbf{1}))\;\wedge$$

$$\forall j : 0 < j \leq i \;\;\Rightarrow\;\; \texttt{nextSectionNumber}(n_{1\bullet}n_{2\bullet}\cdots{}_\bullet n_i,\;\; n_{1\bullet}n_{2\bullet}\cdots{}_\bullet(n_j + 1))$$

Recognizing Characteristic and its Values: The `spec class` designators appear as upper case letters (A, B, C,. . .) along with spec names or keywords such as `class`, though upper case letters can arise as values for hardness, temperature, etc. Relatively speaking, recognition of product and product types is simple, while recognition of alloy names, especially when they are given in terms of their UNS numbers is non-trivial. One can, in general, use regular expressions [17] to specify phrases that express characteristic-value pairs in spec text.

Associating Conditions with Sections/Paragraphs in GE Specs: The conditioned notes can be specified in terms of the scope rules of applicability of charatcteristic-value pairs to the spec text fragments. For instance, to associate a spec class with a section or a paragraph, we use the following heuristic: Every (sub-)section is conditioned on all spec classes named in section 'Scope'. Explicit spec class references in a paragraph override the default condition. Otherwise, a paragraph inherits the condition from its left sibling (earlier paragraph), or transitively from its parent (enclosing (sub-)section). *The rationale behind the heuristic is that, when the conditionals in a Level 1 extraction are evaluated against the given condition values, it should generate all applicable fragments of the spec.*

To abstract the algorithmic details of Level 1 extraction [47, 55], the structure of a spec can be captured using a pseudo context-free grammar (actually, a variant of EBNF)

```
<document> ::= <document-header> <section>+
<section> ::= <sectionNumber> <sectionHeading> <paragraph>+ <section>*
```

and the computation of the conditioned notes involving spec classes can be given using attribute grammar formalism as described in Section 4.3.3. There are many other qualifiers of interest besides spec classes such as products, product types, alloys, etc.

4.3.3 Algorithmic Details : Level 1 Extraction

For spec class computation, there are two fundamental aspects to be addressed: (1) How do we determine spec classes associated with a paragraph that does not contain any explicit reference to spec classes, and (2) What is the extent of spec text to which an explicitly given spec class reference applies. Attribute grammar framework can be used to specify the scope rules. Here, the attributes with `IN`-suffix correspond to the inherited attributes capturing the default values obtained from the surrounding context, while the attributes with the `OUT`-suffix correspond to the synthesized attributes capturing the explicitly given values in the body text. The attribute with `APP`-suffix corresponds to the applicable spec classes obtained using the scope rules such as: (1) Beginning of a section or a subsection ends the scope of the previous spec class reference. (2) A spec class reference prefixing a paragraph is applicable to it. (3) For a paragraph that does not contain any explicit spec class reference, one of the following two possibilites is chosen: (a) It inherits spec classes from the parent section, or (b) It gets spec classes from its previous sibling paragraph.

```
------------------------------------------------------------------------
Nonterminals and the corresponding Inherited/Synthesized Attributes,
which range over set of spec classes (except textIN which is a string):
------------------------------------------------------------------------
<section>  --->  scIN / scOUT, scApp
<sections>, <nestedSections>   --->  scIN / scOUT
<paragraph>   --->  textIN, scIN / scOUT, scApp
<paragraphs>   --->  scIN / scOUT
========================================================================
------------------------------------------------------------------------
Attribute Grammar Productions and Attribute Computation Rules
(Note that ‘U’ denotes the set-union operation and (*...*) denotes comments.)
------------------------------------------------------------------------
<document> ::= <sections>
sections.scIN =  (* Spec classes from Document Section ‘Scope’ *)

<sections> ::= <section>
section.scIN     = sections.scIN
sections.scOUT   = section.scOUT

<sections> ::= <section> <sections_1>
section.scIN     = sections.scIN
sections_1.scIN  = sections.scIN
sections.scOUT   = section.scOUT U sections_1.scOUT

<section> ::= <sectionNumber> <sectionHeading> <paragraphs> <nestedSections>
paragraphs.scIN     = section.scIN
nestedSections.scIN = section.scIN
section.scOUT       = paragraphs.scOUT U nestedSections.scOUT
section.scApp       = section.scIN

<paragraphs> ::= <paragraph>
paragraph.scIN    = paragraphs.scIN
paragraphs.scOUT  = paragraph.scOUT

<paragraphs> ::= <paragraph> <paragraphs_1>
paragraph.scIN    = paragraphs.scIN
paragraphs_1.scIN = left-sibling-defines? paragraph.scOUT : paragraphs.scIN;
paragraphs.scOUT  = paragraph.scOUT U paragraphs.scOUT

<paragraph> ::=  (* text between two blank lines *)
paragraph.scOUT   =  getSpecClasses(paragraph.textIN)
paragraph.scApp   =  explicitly-defines ? paragraph.scOut : paragraph.scIN;
```

```
<nestedSections> ::=  empty
nestedSections.scOUT   = nestedSections.scIN

<nestedSections> ::=  <sections>
sections.scIN          = nestedSections.scIN
nestedSections.scOUT   = sections.scOUT
```

We can now specify generation of conditioned notes, with the gluing of adjacent paragraphs that have the same condition, into one note. For this purpose, we need additional synthesized attributes `FIRST` and `NOTES`, and a string operation `APPEND`. The synthesized attributes `SN` and `SH` stand for section number and section heading, and are used to locate a paragraph.

```
------------------------------------------------------------------------
Nonterminals and the additional Inherited/Synthesized Attributes,
which range over strings (except FIRST which is a set of spec classes):
------------------------------------------------------------------------
<document> ---> / NOTES
<section>  ---> / NOTES
<sections>, <nestedSections>   --->  / NOTES
<sectionNumber> ---> / SN
<sectionHeading> ---> / SH
<paragraph>   --->  SN, SH / NOTES
<paragraphs>   --->  SN, SH / NOTES, FIRST
========================================================================
------------------------------------------------------------------------
Attribute Grammar Productions and additional Attribute Computation Rules
------------------------------------------------------------------------
<document> ::= <sections>
document.NOTES  = sections.NOTES

<sections> ::= <section>
sections.NOTES  = section.NOTES

<sections> ::= <section> <sections_1>
sections.NOTES  = APPEND(section.NOTES, sections_1.NOTES)

<section> ::= <sectionNumber> <sectionHeading> <paragraphs> <nestedSections>
section.NOTES   =   APPEND ("IF section.scAPP THEN",
                           "NOTE: sectionNumber.SN sectionHeading.SH",
                            paragraphs.NOTES, nestedSections.NOTES)
paragraphs.SN   = sectionNumber.SN
paragraphs.SH   = sectionHeading.SH

<paragraphs> ::= <paragraph>
paragraphs.FIRST  = paragraph.scAPP
paragraphs.NOTES  = paragraph.NOTES
paragraph.SN      = paragraphs.SN
paragraph.SH      = paragraphs.SH
```

```
<paragraphs> ::= <paragraph> <paragraphs_1>
paragraphs.FIRST  = paragraph.scAPP
paragraphs.NOTES  = if paragraph.scAPP == paragraphs_1.FIRST
                     then APPEND(paragraph.NOTES,paragraphs_1.NOTES)
                     else APPEND(paragraph.NOTES,
                                 "IF paragraphs_1.scAPP THEN",
                                 "NOTE: paragraphs.SN paragraphs.SH",
                                 paragraphs_1.NOTES)
paragraph.SN     = paragraphs.SN
paragraph.SH     = paragraphs.SH
paragraphs_1.SN  = paragraphs.SN
paragraphs_1.SH  = paragraphs.SH

<paragraph> ::=  (* text between two blank lines *)
paragraph.NOTES   = paragraph.textIN

<nestedSections> ::=  empty
nestedSections.NOTES   = ""

<nestedSections> ::=  <sections>
nestedSections.NOTES   = sections.NOTES
```

To illustrate Level 1 Extraction, consider the following abstracted example showing applicable spec classes and related spec text fragments. The 'Scope'-section is assumed to contain references to spec classes A, B, and C, and other spec class references in the spec body have been made explicit with 'Class' prefix.

```
A,B,C =>       3. General Requirements
A,B,C =>        3.1 Heat Treatment. Para0
A =>                        Class A: Para1
                                       Para2
B =>                        Class B: Para3
A,B,C =>                3.1.1 Elevated Temperature.
A =>                        Class A: Para4
C =>                        Class C: Para5
A,B,C =>                3.1.2 Annealing.
B =>                        Class B: Para6
                                    Class B: Para7
                                       Para8
A,B,C =>       4. Quality Assurance
```

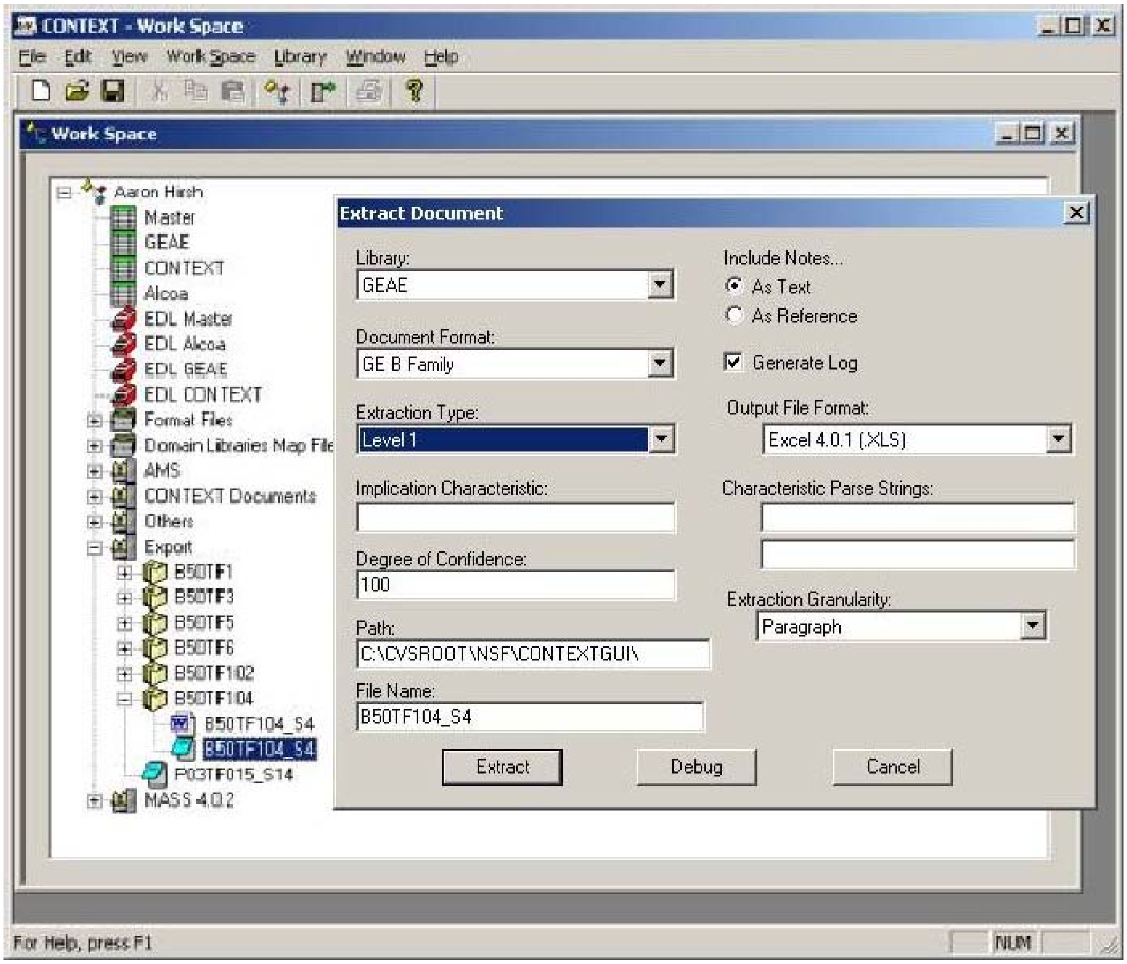
In practice, regular expressions can be used to narrow down the text in which the condition characteristic values are searched.

4.4 Implementation Details

We now describe the Extraction Wizard (called CONTEXT for CONTent EXTractor) that takes a pre-processed spec in plain text form as input and returns the extraction results. The overall approach is to solve the extraction problems by providing a set of tools for flexible handling of specs in different formats and content, and for refining extraction parameters. It deals with issues such as: (1) multiple layouts of spec; (2) use of customized string matching algorithm to enhance domain library terms identification; (3) exporting to different formats for manipulation of the results in various packages; (4) table formatting, image and extended character handling; and (5) tracing the generation and relating extraction results back to the original spec. The user interface provides the extractor with a workbench in which they can perform multiple extractions and tweak extraction parameters. Figure 3 shows the workspace view with defined domain libraries, format files, and multiple storage folders that are used to store specifications and supporting documentation [47, 50]. The actual implementation of the extraction engine has been coded in C++ to interface with the Cohesia's MASS infrastructure support code and in XSLT for tree-traversals over XML data, depending on the application, representation details, and other idiosyncracies of a spec.

Extraction Utility: Extraction utility consists of three main components: the SDR objects, the extraction engine, and the formatter, as depicted in Figure 4. The SDR objects represent the extracted SDR tree. The extraction engine contains the bulk of the functionality and is responsible for parsing the spec text into SDR tree. The formatter outputs the SDR tree in SDR-XML form that can be imported into the Spec Editor for manual massaging and integrated with the existing MASS product, or in XML-annotated Master form that can be further manipulated using generic XML tools [55]. In any case, we still rely on a domain expert to remedy any inadequacies in the automatic markup.

The core of the extraction engine is implemented using four classes: `Extractor`, `Locator`, `DomainLibrary`, and `Tokenizer`. The `Extractor` object is responsible for orchestrating the entire extraction process. The `Config` file is read to determine the extraction type, file path settings, domain library search settings, etc. The `Format` file is consulted to guide the search for spec header information and requirements, and to skip the peripheral text. The detailed extraction is carried out with the help of `Locator` object, which has methods to recognize procedures, characteristics, and controlled terms (that is, domain library terms) appearing in spec text. These methods are implemented on top of the domain library search engine. The `DomainLibrary` object implements advanced domain library search capabilities. It is intended to exploit information about synonyms (e.g., chemistry and composition, etc), prefixes (e.g., electro-, spectro-, etc), suffixes (e.g., -ate, -tion, etc), stop words, broader and narrower terms (e.g., forging and die forging, etc), unit of measure conversions, etc in mapping a spec phrase into semantically equivalent domain library term [53]. In fact, the entire domain library can be viewed as consisting of the *object domain library* (ODL) containing the core set of terms, enclosed by the *English domain library* (EDL) wrapper that incorporates flexible and robust matching algorithm to improve search.



Extract Document

Library: GEAE

Document Format: GE B Family

Extraction Type: Level 1

Implication Characteristic:

Degree of Confidence: 100

Path: C:\CVSROOT\NSF\CONTEXTGUI\

File Name: B50TF104_S4

Include Notes...
● As Text
○ As Reference

☑ Generate Log

Output File Format: Excel 4.0.1 (.XLS)

Characteristic Parse Strings:

Extraction Granularity: Paragraph

Figure 3: Extraction Wizard Interface

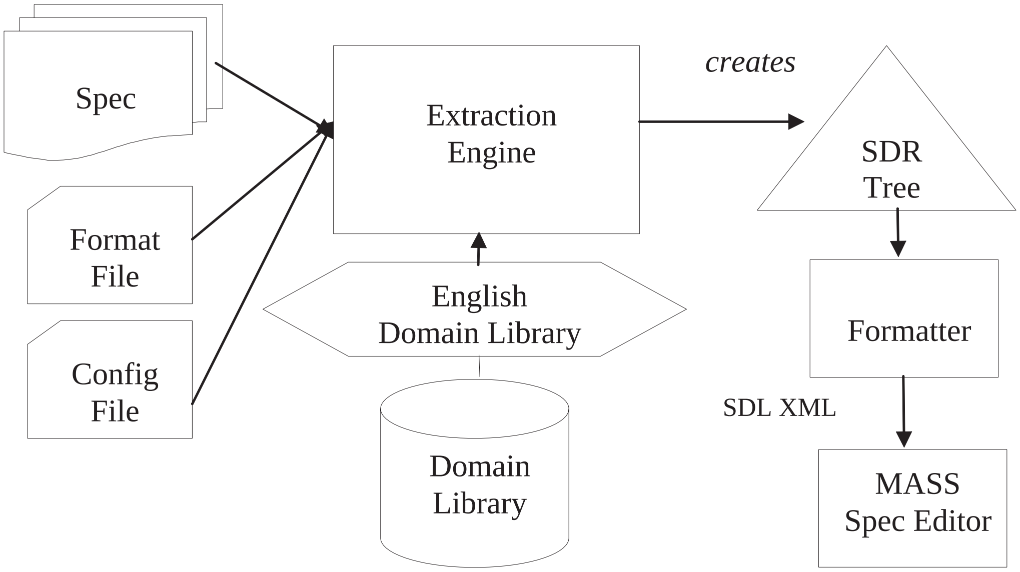


Figure 4: Extraction Engine Architecture

4.4.1 Overall Flow: Design of Software Modules

The syntax of nested sections can be expressed as extended regular expressions, and has been recognized using a lexical analyzer generated by FLEX, and also coded in C++ using RogueWave Libraries. The matching algorithm for phrases has been coded in C++. The Level 1 Extraction has been carried out in four steps:

1. Automatically recognize nested sections of the spec.
2. Automatically recognize spec phrases to be extracted. The spec heading contains information about specification name, revision, products, spec class, alloy name, etc. The spec body contains phrases that refer to domain library terms that can appear in conditions such as products, spec classes, alloy names, etc. Alloy names may appear explicitly or through their standardized UNS number. These are recognized using the appropriate fragment of domain library.
3. Automatically generate multiple views of the XML source using transformations coded in XSLT.
4. Finally, verify, and possibly amend, the resulting translations manually.

4.4.2 Experimental Results and an Example of Level 1 Extraction

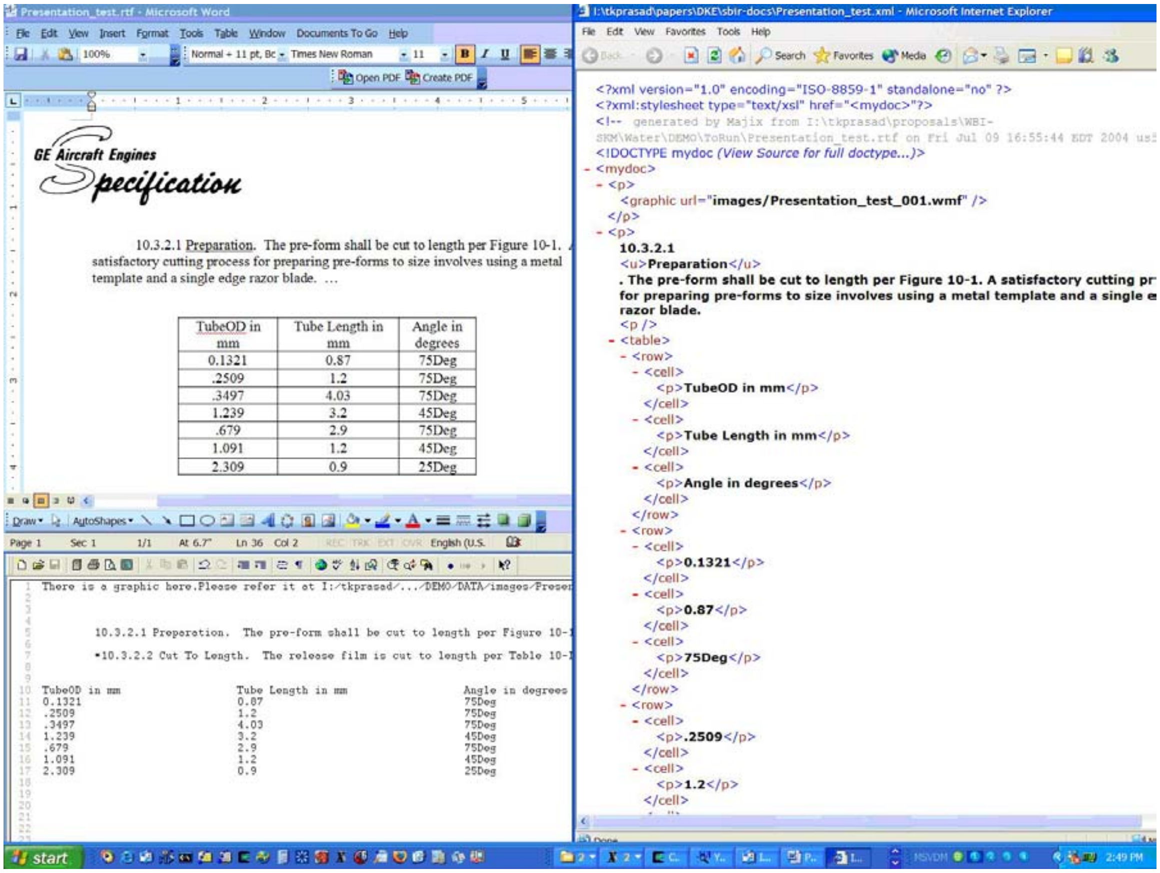
The Materials and Process Specifications from ASTM, SAE, Rolls Royce, Pratt and Whitney, Alcoa, GE, etc are available to us in one of the following formats:

(i) as paper-based hardcopy, (ii) as PDF document, or (iii) as MS WORD document. For each authoring organization, one can compose a separate `Format` file that captures the overall structure of the header information and the location of the requirements to be extracted in the spec body. One can also have multiple domain libraries. The extraction engine is *adaptable* in that one can choose the `Format` file and domain library to use. In general, the structure of the legacy specs and their revisions is fairly stable. Domain libraries can be updated periodically, but the domain library search algorithm is sufficiently robust to deal with the changes.

Conversion of hardcopy specs and PDF specs via scanning and OCR into an electronic form that can be programmatically manipulated turned out to be very unreliable. So our focus turned to GE Aircraft Engines Specs available in MS WORD format. In particular, we had about 300 B-family material specs and about 200 P-family processing specs. The GE domain library contains over 10,000 entries. A spec in MS-WORD format was pre-processed (using Majix and MS-WORD macros) to obtain it in plain text format preserving the text, table layouts, and links to image files. Level 1 Extraction engine works on the plain text document and generates the XML-Master for further manipulation, or for importing into Excel-based Spec Editor for manual inspection and correction. Overall, both the precision and the recall were high, that is, in the range of 90% to 100% depending on the input spec. In practice, a domain expert usually reviews the final extraction results. In addition to correcting extraction errors, they analyzed errors and provided constructive feedback to software engineers for improving the performance of the next revision of Level 1 Extraction engine. It is instructive to consider, as an illustrative example, the manifestation of spec class designator in GE Specs. Typical spec class references are upper case letters (A, B, C,. . .) that appear as in: 'B50T75A', 'B50T75A, E, F and G', 'Class G, H, and I', 'Classes A and C', 'P21TF7 Cl-A', etc. However, upper case letters can also appear in other contexts, such as a spec reference 'ASTM E 46' or as a (hardness) value such as 'Rockwell B 40-75', or for unit of temperature (F or C), etc. Further, modifiers such as 'similar to' preceding a spec class designator can imply that the designator is not playing the role of a condition.

Now we consider concrete examples. Figure 5 is a screenshot depicting: (i) a fragment of GEAE Spec containing text, a table, and an image opened in MS WORD, (ii) the XML intermediary generated by Majix opened in Internet Explorer 6, and (iii) the corresponding plain text file opened in TextPad.

The spec in plain text form is subsequently tagged to capture and make explicit the structure and the content to different levels of detail. The XML-Master document embodies the literal translation strategy by embedding extracted fragments into the spec. XSLT Stylesheets are then employed to filter and transform the XML-Master document into suitable output forms using Apache's Xalan processor. In the context of Level 1 Extraction, the views of interest are: the original text view with identation, for verification; the HTML view for display in a Web-browser; generation of syntactically correct SDR with the paragraph text presented as conditioned notes; and generation of syntactically correct SDR with the paragraph presented as conditioned notes using



Presentation_test.rtf - Microsoft Word
File Edit View Insert Format Tools Table Window Documents To Go Help
100%
Normal + 11 pt, Bc
Times New Roman
11
Open PDF
Create PDF
GE Aircraft Engines
Specification
Draw
AutoShapes
Page 1 Sec 1 1/1 At 6.7" Ln 36 Col 2 REC TRK EXT OVR English (U.S.
I:\tkprasad\papers\DKE\sbir-docs\Presentation_test.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Done
start
2:49 PM

10.3.2.1 Preparation. The pre-form shall be cut to length per Figure 10-1. A satisfactory cutting process for preparing pre-forms to size involves using a metal template and a single edge razor blade. …

TubeOD in mm	Tube Length in mm	Angle in degrees
0.1321	0.87	75Deg
.2509	1.2	75Deg
.3497	4.03	75Deg
1.239	3.2	45Deg
.679	2.9	75Deg
1.091	1.2	45Deg
2.309	0.9	25Deg

```
There is a graphic here.Please refer it at I:/tkprasad/.../DEMO/DATA/images/Presen



       10.3.2.1 Preparation.  The pre-form shall be cut to length per Figure 10-1

       •10.3.2.2 Cut To Length.  The release film is cut to length per Table 10-1


TubeOD in mm                 Tube Length in mm                Angle in degrees
0.1321                       0.87                             75Deg
.2509                        1.2                              75Deg
.3497                        4.03                             75Deg
1.239                        3.2                              45Deg
.679                         2.9                              75Deg
1.091                        1.2                              45Deg
2.309                        0.9                              25Deg
```

```
  <?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
  <?xml:stylesheet type="text/xsl" href="<mydoc>"?>
  <!-- generated by Majix from I:\tkprasad\proposals\WBI-
  SKM\Water\DEMO\ToRun\Presentation_test.rtf on Fri Jul 09 16:55:44 EDT 2004 us
  <!DOCTYPE mydoc (View Source for full doctype...)>
- <mydoc>
  - <p>
      <graphic url="images/Presentation_test_001.wmf" />
    </p>
  - <p>
      10.3.2.1
      <u>Preparation</u>
      . The pre-form shall be cut to length per Figure 10-1. A satisfactory cutting pr
      for preparing pre-forms to size involves using a metal template and a single e
      razor blade.
      <p />
    - <table>
      - <row>
        - <cell>
            <p>TubeOD in mm</p>
          </cell>
        - <cell>
            <p>Tube Length in mm</p>
          </cell>
        - <cell>
            <p>Angle in degrees</p>
          </cell>
        </row>
      - <row>
        - <cell>
            <p>0.1321</p>
          </cell>
        - <cell>
            <p>0.87</p>
          </cell>
        - <cell>
            <p>75Deg</p>
          </cell>
        </row>
      - <row>
        - <cell>
            <p>.2509</p>
          </cell>
        - <cell>
            <p>1.2</p>
          </cell>
```

Figure 5: MS Word to Plain Text via XML

paragraph numbers in place of paragraph bodies, to deal with copyright restrictions that prohibit duplication of the spec text verbatim in another document.

A fragment of the original GE B-family spec B50TF104 in text form, `GE.txt`, is shown below.

```
            SPECIFICATION NO.  B50TF104
                    ISSUE NO.  S4+AM1
                         DATE  January 9, 1996
                    PAGE 1 OF  10
                    CAGE CODE  07482
                   SUPERSEDES  B50TF104-S3
...
ALLOY BAR, FORGINGS, AND RINGS
(INCONEL ALLOY 706)
...
3.2.1  Material Condition.  Material shall be delivered in the following
condition specified on the purchaser order:

(a) Bar and rod shall be hot finished, heat treated per CLASS A, and
descaled; round bars shall be ground or turned.  Forgings shall be
as ordered.

(b) Flash welded rings shall not be supplied unless specified or
permitted on Purchaser’s part drawing.
...
```

`GE.txt` is tagged with respect to structure and content for Level 1 Extraction to obtain `GE.xml` as shown below. (This reproduction contains additional newlines to make the XML text not overflow the right margin.)

```
<?xml version="1.0"?>
<document>
            SPECIFICATION NO.<specNumber>B50TF104</specNumber>
                    ISSUE NO.<issueNumber>S4+AM1</issueNumber>
                         DATE  January 9, 1996<nl/>
                   PAGE 1 OF  10<nl/>
                    CAGE CODE  07482<nl/>
                   SUPERSEDES  B50TF104-S3<nl/>
...<nl/>
ALLOY <product DLT="Bar">BAR</product>, <product DLT="Forgings">FORGINGS</product>,
AND <product DLT="Rings">RINGS</product><nl/>
(INCONEL ALLOY 706)<nl/>
...<nl/>
<section level="4" id="3.2.1">   3.2.1
  <sectionHeading heading="Material Condition">Material Condition.</sectionHeading>
Material shall be delivered in the following <nl/>
condition specified on the purchaser order:<nl/>
(a) <product DLT="Bar">Bar</product> and <product DLT="Rod">rod</product> shall be hot finished,
heat treated per <specClass DLT="A">CLASS A</specClass>, and <nl/>
descaled; round <product DLT="Bar">bars</product> shall be ground or turned.
<product DLT="Forgings">Forgings</product> shall be <nl/>
as ordered.<nl/>
(b) Flash welded <product DLT="Rings">rings</product> shall not be supplied unless specified or <nl/>
permitted on Purchaser’s part drawing.<nl/>
</section>
</document>
```

`GE.xml` is then transformed into the following Level 1 Extraction, with respect to `products` and `spec classes` and with spec text suppressed, in `GE.sdr` using a suitable XSLT stylesheet.

```
document [B50TF104]
```

```
{
  title = "ALLOY BAR, FORGINGS, AND RINGS(INCONEL ALLOY 706)";
   org  = "GE Aircraft Engines";
   type = "specification";
  define APM
  {
       [Products] is "Bar";
       [Products] is "Forgings";
       [Products] is "Rings";
  }
  using APM;

  revision [S4+AM1]
  {
...
  if
      ( [Product Type] is "Bar"  or [Product Type] is "Rod"  or
          [Product Type] is "Forgings"  or [Product Type] is "Rings" )  and  ( [Spec Class] is "A" )
    then
    {
         note " = Material Condition."
         "Shall be in accordance with paragraph 3.2.1  "
    }
...
  }
}
```

To conserve space, we are omitting other possible translations obtainable from the XML-Master by applying algorithms coded as XSLT stylesheets.

5 Conclusion

Semi-structured documents are ungrammatical texts that have recognizable organization and constrained vocabulary. These arise naturally in the context of technical specifications of materials and processes, and are crucial to companies involved in complex manufacturing and B2B E-commerce. In this work, we developed computer-assisted coarse-level content extraction tools. It was a challenge to scope the problem to make it tractable to the software developers while simultaneously ensuring that the results are useful in unburdening the domain experts.

Specifically, we attempted to systematically evolve the manual extraction tool into a computer-assisted extraction tool. We carefully analyzed specs to understand and explicate their nature, and studied extractions from specs – of different complexities and origin – carried out by highly trained metallurgists, and appreciated the importance of making domain library searches flexible. The recognition of semantically equivalent phrases that map to the same Domain Library Term was attempted using a normalization procedure starting from verbatim searches, to using minor variations on words, to using aliases, to eventually using much richer patterns of equivalences.

For a semi-automatic approach to succeed in practice, the partial results obtained through automation should be intelligible, in relation to the original spec. Otherwise, the extractors will have a difficult time completing the extraction. Thus, semi-automatic approach benefited from maintaining a one-to-one correspondence between the spec and its translation. The XML Technology seems appropriate to extraction to the extent that the XML Master provides a way of tying together the spec and the extracted information, and the use of XSLT stylesheets in transforming this information as desired by the context.

Extraction Wizard embodies a delicate compromise between the desired flexibility with respect to the input/output formats and transparency of operation with the option of single-stepping through the extraction process using a special debugger, and what can be realistically programmed. Based on our first experiences with the prototypes we have built, the generated Level 1 extractions seem reasonable though not perfect from domain expert's perspective. Extraction wizard also handles heterogeneous spec documents by separating text from images and tables, so that text can be processed through the extraction engine while deferring tables, footnotes, and images for manual handling.

The development of CONTEXT brought together tools and techniques from Language Processing, Knowledge Representation, and Web Technologies. What we have accomplished are the necessary initial steps towards automatic reorganization and summarization of semi-structured documents.

Acknowledgment

The authors would like to thank Steve Grace, Steve Crowley and Prasanna Soundarpandian for numerous discussions throughout the project.

References

[1] N. Ashish and C. A. Knoblock: Semi-automatic wrapper generation for Internet information sources, In *Proceedings of the Second IFCIS International Conference on Cooperative Information Systems*, Kiawah Island, SC, 1997.

[2] R. Baumgartner, S. Flesca, and G. Gottlob: Visual Web Information Extraction with Lixto, In: *27th International Conference on Very Large Databases*, pp. 119-128, 2001.

[3] T. Berners-Lee, J. Hendler, and O. Lassila: The Semantic Web, Scientific American, May 2001 issue. (http://www.sciam.com/)

[4] S. Brin: Extracting Patterns and Relations from the World Wide Web. In: *WebDB Workshop at 6th International Conference on Extending Database Technology*, 1998.

[5] M. Califf and R. J. Mooney: Relational Learning of Pattern-Match Rules for Information Extraction, *Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pp. 6-11, 1998.

[6] V. Crescenzi, G. Mecca, and P. Merialdo: RoadRunner: Towards Automatic Data Extraction from Large Web Sites, In: *27th International Conference on Very Large Databases*, pp. 109-118, 2001.

[7] A. Crespo, J. Jannink, E. Neuhold, M. Rys, and R. Studer: A Survey of Semi-Automatic Extraction and Transformation, *Information Systems*, 2000.

[8] M. C. Daconta, L. J. Obrst, and K. T. Smith: *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, John Wiley and Sons, 2003.

[9] B. Du Charme: *XSLT Quickly*, Manning Publications Co., 2001.

[10] D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle: Ontology-based extraction and structuring of information from data-rich unstructured documents, Proceedings of the 7th International Conference on Information and Knowledge Management, pp. 52 - 59, 1998.

[11] F. van Harmelen and D. Fensel: Practical Knowledge Representation for the Web. In *Proceedings of the Workshop on Intelligent Information Integration (III99)* , 1999.

[12] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, and A. Witt: On2broker: Semantic-based access to information sources at the WWW, In *Proceedings of World Conference on the WWW and Internet (WebNet99)*. 1999.

[13] D. Fensel: Semantic Web enabled Web Services, Invited Talk at: In *7th International Workshop on Applications of Natural Language to Information Systems*, June 27-28, 2002, Stockholm - Sweden.

[14] D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster (Editors): *Spinning the Semantic Web : Bringing the World Wide Web to Its Full Potential*, MIT Press, 2003.

[15] D. Fisher, S. Soderland, S. McCarthy, F. Feng and W. Lehnert: Description of the UMass Systems as Used for MUC-6, *Proceedings of MUC-6*, pp. 221-236, 1996.

[16] D. Freitag: Information extraction from HTML: application of a general machine learning approach, *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pp. 517-523, 1998.

[17] J. Friedl, *Mastering Regular Expressions*, 2nd Edition, O'Reilly, 2002.

[18] Fujii, A., and Ishikawa, T.: Cross-Language Information Retrieval: Exploration of Query Translation and Transliteration. Computers and the Humanities, Vol.35, No.4, pp. 389-420, 2001

[19] `http://gate.ac.uk/`

[20] R. Grishman: Information Extraction: Techniques and Challenges, Information Extraction (International Summer School SCIE-97), ed. Maria Teresa Pazienza, Springer-Verlag, 1997.

[21] R. Grishman: The NYU System for MUC-6 or Where's the Syntax?, In *Proceedings of the Sixth Message-Understanding Conference (MUC-6).* San Francisco, CA, Sundheim, B., and Grishman, R., eds., Morgan Kaufmann. 1995.

[22] T. R. Gruber: Toward principles for the design of ontologies used for knowledge sharing, Originally in: *International Workshop on Formal Ontology*, Padova, Italy. Later in: *International Journal of Human-Computer Studies*: Special issue: Formal Ontology in Conceptual Analysis and Knowledge Representation, Guarino N. and Poli R. (Eds), Kluwer. 1993.

[23] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha: Extracting Semistructured Infromation from the Web, In Proceedings of the Workshop on Management of Semistructured Data, 1997.

[24] J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson: FASTUS: Extracting Information from Natural-Language Text, 1996. `http://www.ai.sri.com/natural-language/projects/fastus-schabes.html`

[25] D. Hull: Stemming Algorithms: A Case Study for Detailed Evaluation, *Journal of the American Society for Information Science*, Vol. 47(1), pp. 70-84, 1996.

[26] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, J. S. Teixeira: A Brief Survey of Web Data Extraction Tools. SIGMOD Record 31(2): 84-93 (2002)

[27] W.G. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland: Evaluating an Information Extraction System, *Journal of Integrated Computer-Aided Engineering*, 1(6) pp. 453-472, 1994.

[28] J. McCarthy: *A Trainable Approach to Coreference Resolution for Information Extraction*, PhD Thesis. Dept. of Computer Science Technical Report # 78, University of Massachusetts, Amherst.

[29] Proceedings of the Third Message Understanding Conference, Morgan Kaufman, 1991.

[30] Proceedings of the Fourth Message Understanding Conference, Morgan Kaufman, 1992.

[31] Proceedings of the Fifth Message Understanding Conference, Morgan Kaufman, 1993.

[32] Sixth Message Understanding Conference, 1996. `http://cs.nyu.edu/cs/faculty/grishman/muc6.html`

[33] Seventh Message Understanding Conference, 1997. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings /muc_7_toc.html`

[34] I. Muslea: Extraction Patterns for Information Extraction Tasks: A Survey, In *Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[35] I. Muslea, S. Minton, and C. Knoblock: Hierarchical Wrapper Induction for Semistructured Information Sources, *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93-114, 2001.

[36] L. Obrst, K.N. Jha, and G. Coen. Mass Change of On-line Textual Databases Using Natural Language Processing, In *Proceedings of The 9th Symposium and Exhibition on Industrial Applications of Prolog*, INAP '96, Tokyo, Japan.

[37] L. Obrst and K.N. Jha, NLP and Industry: Transfer and Reuse of Technologies, In *From Research to Commercial Applications: Making NLP Work in Practice*, Eds: Jill Burstein and Claudia Leacock, Association for Computational Linguistics, pp. 57-63, 1997.

[38] Ontoweb: An Ontology-based information exchange for knowledge management and electronic commerce. `http://ontoweb.aifb.uni-karlsruhe.de/`

[39] C. D. Paice: Another stemmer, *SIGIR Forum*, 24(3), pp. 56-61, 1990.

[40] C. D. Paice: An evaluation method for stemming algorithms, In: *Proceedings of the 15th ACM SIGIR conference*, Croft, W.B. and van Rijsbergen, C.J. (eds.), pp. 42-50, 1996.

[41] M. F. Porter: An Algorithm for Suffix Stripping, *Program*, Vol. **14**, No. 3, pp. 130–137, 1990.

[42] B. Ribeiro-Neto, A. H. F. Laender, and A. S. da Silva: Extracting semistructured data through examples, Proceedings of the 8th International Conference on Information and Knowledge Management, pp. 94 - 101, 1999.

[43] E. Riloff: Automatically Constructing a Dictionary for Information Extraction Tasks, In *Proceedings of the Eleventh Annual Conference on Artificial Intelligence* pp. 811-816, 1994.

[44] *Readings in Information Retrieval*, Edited by Karen Sparck Jones and Peter Willett, Morgan Kaufmann, 1997.

[45] D. Z. Sokol: Concurrent Engineering in the Materials Industry: Case Study in the Application of Information Technology, *Fourth Annual Conference on Management of Technology*, 1994.

[46] D. Z. Sokol and J. Rowe: Integrating STEP and SGML for Concurrent Engineering, CALS 95 International Expo.

[47] D. Z. Sokol et al: Computer Assisted Document Interpretation Tools, NSF Phase II Final Report (Executive Summary and Technical Details), 2002.

[48] S. G. Soderland: CRYSTAL: Learning Domain-specific Text Analysis Rules, CIIR Technical Report # 43, University of Massachusetts at Amherst.

[49] S. G. Soderland: Learning Information Extraction Rules for Semi-structured and Free Text, *Machine Learning*, Vol. **34**, No. 1-3, pp. 233–272, 1999.

[50] P. Soundarapandian: *Information Extraction for Reorganizing Materials and Process Specifications*, M.S. Thesis, Dept. of Computer Science and Engineering, Wright State University, 2002.

[51] `http://www.semanticweb.org/`

[52] `http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/`

[53] K. Thirunarayan, A. Berkovich, and D. Z. Sokol, Semi-automatic Content Extraction from Specifications, In *Proceedings of International Workshop on Applications of Natural Language to Information Systems*, LNCS/LNAI, Springer Verlag, 2002.

[54] K. Thirunarayan, Applying XML/XSLT Technology : A Case Study, In *Proceedings of International Symposium on Information Systems and Engineering*, Simulation Series Vol 34(2), pp. 73-78, 2002.

[55] K. Thirunarayan, A. Berkovich, S. Grace, and D. Z. Sokol, Information Extraction for Reorganizing Specifications, *Proceedings of International Workshop on Applications of Natural Language to Information Systems*, LNI, Springer Verlag, pp. 242-248, 2003.

[56] D. Tidwell: *XSLT*, O'Reilly, 2001.

[57] R. Yangarber and R. Grishman: Description of the Proteus/PET System used for MUC-7, *Proceedings of the MUC-7*, 1998.