UNIVERSITY OF
EASTERN FINLAND

University of Eastern Finland

School of Computing

Master's Thesis

Extending CoProE – An Ontology for Storing and Acquiring Business Intelligence

Tabish Fayyaz Mufti

March 2012

ABSTRACT

This thesis reports on the extension of *company, product and event* (CoProE) ontology, an ontology for storing and acquiring *Business Intelligence* (BI) that was developed as part of larger *text mining* (TM) system for collecting and analyzing business texts. We present the concepts behind creating ontologies and the internal composition of CoProE. The TM system needs an intelligent data store environment for its *domain knowledge base* from where it can efficiently retrieve business-related data from text documents and manipulate different parameters for supporting business decision-making. As part of this thesis, we collected and inserted data about five Finnish companies and their products into CoProE using ontology-editing software Protégé. The business information was collected both from public sources as well as by conducting a survey. Furthermore, in order to have the ability to programmatically store and retrieve information about business entities and events into and from CoProE, we developed a Java *Application Programming Interface* (API) for CoProE. Finally, we evaluated the API by using established testing processes. Applying the methodology utilized in this thesis to other BI system has potential for further understanding of the individualities of developing efficient knowledge systems for future intelligent BI systems.

D.1.5 Object-oriented Programming
H.3 INFORMATION STORAGE AND RETRIEVAL
I.2.4 Knowledge Representation Formalisms and Methods
J.1 ADMINISTRATIVE DATA PROCESSING

ACKNOWLEDGEMENTS

I am very grateful to the University of Eastern Finland for giving me the opportunity to come and participate in the International Master's Degree Programme in Information Technology (IMPIT). I am also thankful to the Finnish government for providing me an opportunity to come, study and explore Finland.

This work was supported by the "Towards e-leadership: higher profitability through innovative management and leadership systems" project, which is funded by the European Regional Development Fund and Tekes – the Finnish Funding Agency for Technology and Innovation. I would like to sincerely thank Prof. Erkki Sutinen for introducing me to the "Towards e-leadership" project and without whom I would not have been part of the project. I would extend my greatest thanks to Dr. Tuomo Kakkonen for his continuous help and guidance through out the research work. His supervision during the thesis writing process has been invaluable. The thesis wouldn't have been possible without his support.

I would like to express my thanks to all the friends and acquaintances that provided social support and encouragement during my masters program and stay in Joensuu.

Helsinki, Finland, March 23rd, 2012
Tabish Fayyaz Mufti

ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
BI	Business Intelligence
CoProE	Company, Product and Event Ontology
DAVID	Data Analysis and Visualization aID
IDE	Integrated Development Environment
IE	Information Extraction
MUSING	MUlti-industry, Semantic-based next generation business INtelliGence
OI	Ontology Interfacer
OWL	Web Ontology Language
RDF	Resource Description Framework
SCEO	Sub-Classed Event Objects
SDK	Software Development Kit
SPARQL	SPARQL Protocol and RDF Query Language
SW	Semantic Web
TM	Text Mining
UNSPSC	United Nations Standard Products and Services Code
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

TABLE OF CONTENTS

1 Introduction

This thesis reports on the extension of *company, product and event* (CoProE) ontology, an ontology for storing and acquiring *Business Intelligence* (BI). CoProE was developed as part of larger *text mining* (TM) system for collecting and analyzing business texts [7]. Ontology is an advanced form of data representation that is built from *Semantic Web* (SW) constructs. BI refers to techniques and tools that aim to provide businesses support for better business decision-making by collecting and analyzing relevant information. Business-related events form the core knowledge stored in the TM system. The TM system utilizes business resources (i.e. company entities, product entities) that were inserted in CoProE, to search for business events from business information sources and in response store them into CoProE

As part of this thesis, we collected and inserted data about five Finnish companies and their products into CoProE using ontology-editing software. The data was collected both from public sources as well as by conducting a survey. Further more, in order to have the ability to programmatically store and retrieve business events into CoProE we developed a Java *Application Programming Interface* (API) for CoProE. The ontology population work and the API implemented for this thesis are of fundamental importance for the development of the TM system; it allows the system to leverage the information stored in CoProE for carrying new research or innovating new systems for BI.

1.1 Motivation

World Wide Web (WWW) is the largest information system known to mankind and the amount of information that has been digitized has increased at an incredible rate over the last few decades. As of May 2009 the amount of data available in digital format was roughly estimated to be 500 exabytes [1] (1 exabyte = 1 billion gigabytes). By the year 2010 the amount of data was projected to reach 25 zettabytes [2] (1 zettabyte = 2^{10} exabyte). The web has changed the use of computers from mere numerical devices to a social tool that people use to seek products, social relationships, information, gaming, and news among other things.

This information surge and creative ways of its utilization is leading towards a drive termed *Semantic Web* (SW). The *World Wide Consortium* (W3C), an international standardization body for the Web, propagates the SW. SW was started in late 1990s. The origin of this initiative is attributed to Tim Berners Lee [3], the very person who invented WWW in the late 1980s. SW is gaining relevance and popularity as more and more users generate real time content with the help of powerful tools such as blogs, wikis, social networking tools, news aggregators, video-sharing and shopping sites.

Linguistically the word 'semantic' refers to study of meaning that is used by humans to express them through language [4]. In computer science, the word 'semantic' means unambiguous [2]. What we refer to when saying that we make the web semantic is that we want to know exactly what is the meaning of particular web content and any software that encounters the web also exactly know what it means [2]. The word 'Apple', for example, could refer to a fruit or a company. Search engines, such as Google and Yahoo are one of the most powerful software agents on the web. However, it is appropriate to say that the present search engines are used as information *location-finders*; SW-based search is about transforming the web into an information *retrieval* system. The shift is from keyword-based searching to real time query answering. According to Google CEO Eric Schmidt's vision of the future, Google will search for you even before you ask [5].

Most information is available in weakly structured form e.g. text, audio and video. There are limitations in searching, extracting, maintaining, uncovering, aggregating and viewing information. The effort to overcome these limitations is where the need for SW arises. While the original web initiative has been about linking documents to documents, SW is about linking data to data, regardless of whether particular web content is formatted for human readers or for software agents.

The topic of the thesis is also connected with SW. The thesis is part of a larger research project entitled *"Towards e-leadership: higher profitability through innovative management and leadership systems"*. One of the main aims of the

project is to develop a TM system for text documents that helps business leaders in gathering and analyzing data for BI, customer opinion and feedback analysis as well as for aiding decision-making. Due to the amount of data being digitized at an alarming rate there is a large amount of business information that is being generated on the Internet on a daily basis. The business community needs to take advantage of these information sources to stay competitive in the market. BI refers to techniques and tools that aim to provide businesses with support for better business decision-making by collecting and analyzing business information. TM refers to the automatic process of deriving high quality previously unknown information from text [6]. CoProE ontology provides the business knowledge base as semantic data that the TM system aims to leverage. For that purpose, we populate CoProE with relevant business data and provide a Java API for accessing and storing data into CoProE.

1.2 Research Problem

The CoProE ontology is an ontology that was developed for use as a component of the DAVID (Data Analysis and Visualization aID) system that is developed in the "Towards e-leadership: Higher profitability through innovative management and leadership systems" project [7] that is funded by the European Union and Tekes - Finnish funding agency for technology and innovation. In order to be able to find relevant BI, the system requires information about the companies that are being analyzed as well as their competitors and products. This *background knowledge* is stored in CoProE ontology and forms the basis on which the TM system can locate, analyze and provide decision-making information to the system user.

The system is targeted to diverse companies with different BI requirements. It is, hence, important to validate flexibility of ontology so that it caters to the analysis requirements of various companies otherwise the ontology design would need readjustment. The extent of knowledge to be stored in the ontology requires further research. The ontological data about companies and their products forms the basis for monitoring customer opinion, a significant market event or a change in a particular company's patents. The DAVID system needs business

resources (i.e. company entities, product entities) for searching relevant market events from data sources and it needs structured access to the gathered data to process it effectively. It requires a programming interface so it can manipulate the ontology and store or modify existing events into it. They form the core of business knowledge in the ontology. The importance of market events for the system signifies the requirement for programmatic access to CoProE.

The collection of data about companies and their products, building of a programming interface needs to be carried out under a suitable research design. These mentioned points form the fundamental point of exploration that is to be performed as part of the thesis.

1.3 Research Objectives

Following research objectives have been identified as part of the master thesis:

1. Collect data about companies, products and related information to populate the CoProE ontology.

2. Modify CoProE in case it is not capable to store gathered business data

3. Design and implement a Java API for searching and manipulating CoProE

4. Verify the working of the API and test it under a suitable methodology

The population of CoProE (Objective 1,2) can be published on the web as new business ontology. TM systems could leverage CoProE for conducting new research or innovating new systems for discovering and analyzing BI. Ontology reuse is cost-effective and can greatly reduce the implementation time and effort for a new system. The knowledge stored in the ontology forms the basis for business analysis and as more information is digitized the business community would be forced to utilize latest technologies to stay competitive in the market. In effort to populate the ontology with relevant data most useful sources of company and products data will be highlighted. The development of Java API

(Objective 3,4) will serve as a guide for practitioners to develop a programmable interface on top of business oriented semantic data.

1.4 Research Methods

Two significant outcomes dictated the research methods for the project. First significant outcome was the collection of data about participating companies and their products. Second outcome was the implementation of Java API to store and manipulate data through CoProE – business ontology of the TM system. The data collection was performed through questionnaires and content analysis available about companies on Internet or through information published by companies themselves. It is to be noted that any information shared by companies was under a non-disclosure agreement and it cannot be disclosed to a third party.

For the development of Java API we took a constructive research approach and used the semantic business data as a pivotal guideline for the design of the API. In the design of the API, we took into consideration how it could be built in a way that it stays scalable and adaptable to the changes in business data and the DAVID system. As Java is an object-oriented programming language it enabled us to reuse code and provide uniform interface for different types of business data. An open source third party Java library was utilized to interface with CoProE ontology. The details of API construction from design to final artifact and its evaluation have been documented in the thesis.

1.5 Structure of the Thesis

The thesis is divided into five chapters. The motivation, underlying concepts, research objectives and methods to tackle the research problem are introduced in Chapter 1. Chapter 2 describes the background of the research problem. Chapter 3 discusses in detail the data required for the research, data collection methodology and characteristic of gathered data. The purpose of Chapter 4 is to describe the format and nature of business data that requires an API for interacting with the CoProE ontology. It further discusses the implementation details of the API and the Java SW framework that was used for development. The final chapter contains analysis of the research performed, the conclusions and directions for future work.

2 Background

2.1 Semantic Web Technologies

As described in the previous section, SW is an extension of the current web in which web data is given well defined meaning; better enabling computers and humans to work together. The WWW framework can be understood as based on two content approaches: *push*-content and *pull*-content. In push-content approach content is pushed to us in a certain time and format where as in pull-content approach content is much more individual and contextualized. The user decides what information is wanted, when and how (e.g. TV versus YouTube).

While the existing WWW framework is primarily based on the push-content approach, the web is transitioning towards a pull-content approach. It is the SW that makes this transition possible at world scale [2]. According to the estimate by David et al. [8] by 2012, 80% of public websites will use some level of semantic hypertext to create SW documents and 15% will use more extensive SW-based ontologies to create semantic databases. There is already plenty of semantic data available on the web. For example, BestBuy exposes it as GoodRelations [9], that is the most powerful language for embedding product, price, and company data into existing web pages, Facebook as OpenGraph [10] to allow any webpage to become a rich object in a social graph, Twitter as annotations [11] so that any metadata can be connected to any Twitter message and LinkedIn as MicroFormat [12]. Microformats enable the encoding and extraction of events, contact information, social relationship and so on.

2.2 Ontologies

SW is not a new technology rather it builds on the existing web and presents it in languages specifically designed for information: *Resource Description Framework* (RDF), *Web Ontology Language* (OWL), and *Extensible Markup Language* (XML). XML is a flexible text format for developing structured computer documents in machine-understandable form. It provides a very flexible way to describe complex data such as invoices, news feeds, inventory data etc. RDF is built upon XML and provides a more meaningful way of accessing data from Internet resources. It can provide metadata for web pages making searching more

capable. OWL is build on RDF and adds more vocabulary for describing properties and classes, relation between classes, characteristics of properties etc.

Ontologies leverage the three languages introduced above and provide the means for modeling concepts, attributes and relationships in a specific domain. Ontology composes of the following constructs to store and represent knowledge: *Class*, *Property*, *Individual, Relationship*, *Constraint* and *Axiom*. A *Class* is like a type, kind or category of existing things in the real world. A *Property* can be used to describe a *Class* by linking it to another *Class*. *Individual* is instance or object of a class; it is the basic component of ontology and *Relationship* defines the way a class can be related to an *Individual*. A *Constraint* is a formal rule that describes what must be true about a *Property* or *Relationship*. The last construct *Axiom* is a logical rule and it forms the basis of inference [13]. A SW solution uses ontology as a form of knowledge representation (or data representation) within a certain domain. Figure 1 provides a simple example of ontology.

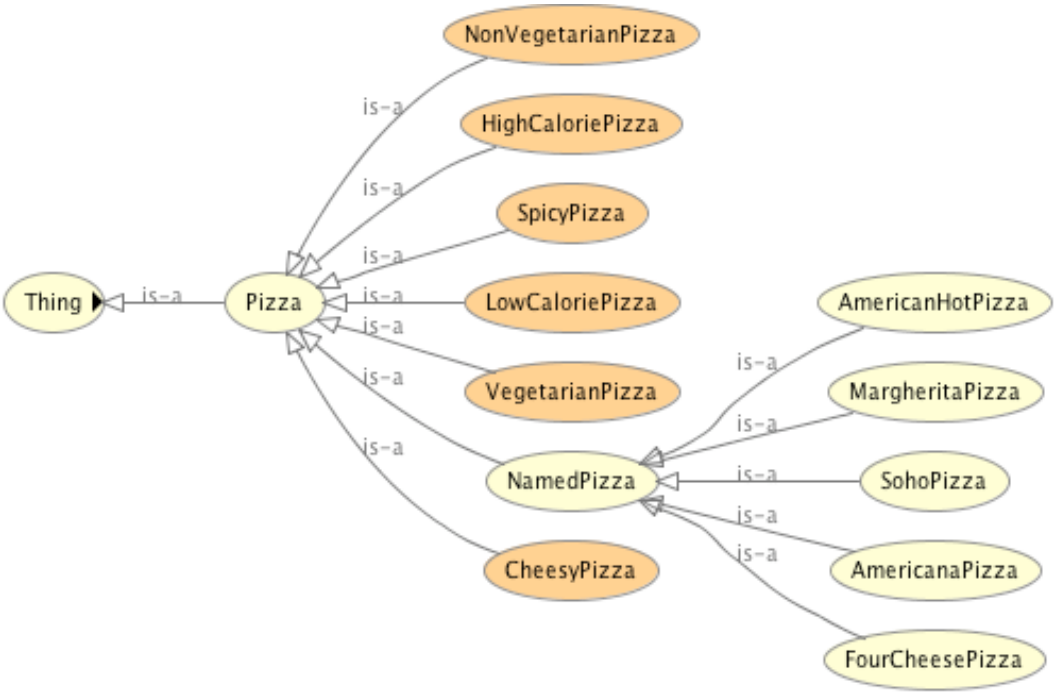

Figure 1 - Example of Pizza stored as Ontology

The ontology illustrated in Figure 1 defines a hierarchy of types of pizzas that are available in a restaurant. Pizza, can for instance, be of the type *Non-Vegetarian* or *Vegetarian*. A *NamedPizza* can further encapsulated under different type of

category. One way of storing pizza information in ontology is through RDF language using an *RDFTriplet*. An *RDFtriplet* is of the form (subject, predicate, object). We can think of this triple (*x*, P, *y*) as a logical formula P(*x* , *y*), where the binary predicate P relates the object *x* to the object y e.g. (*PizzaX*, isType, *Vegetarian*).

2.3 CoProE Ontology

CoProE is based on two constructs: the *newsEvents ontology* [14] and *United Nations Standard Products and Services Code* (UNSPSC) [15] classification of products and segments of industries. The newsEvents ontology is being developed at the University of Karlsruhe for modeling business events, the affected business entities and the relations between them. The development of the ontology is a part of an effort to create an event detection system for BI. The ontology is freely available in OWL format. UNSPSC is a hierarchically organized coding system for classifying products and services that aims at being "an open, global multi-sector standard for efficient, accurate classification of products and services"[15]. The merger of these two constructs leads to a well-classified and detailed structure for CoProE. More details regarding the development of CoProE and its relationship with newsEvent and UNSPC are given in [16].

CoProE has the provision to store knowledge about business events, companies, their products and services on offer as well as relationship with competitors companies and products. CoProE provides a hierarchy of classes for storing business information. Any object or subject of an *RDFTriplet* inserted into CoProE is called a *resource* and a *class* defines the type of a resource. This hierarchy and what each class within this hierarchy can potentially subsume are briefly summarized in Table 1.

Table 1 - Class hierarchy in CoProE

Class	**Examples of what class can subsume**
Entity	*Any resource that is a real world entity*
Place	Resource that is a physical place
Country	Resource that is a country
Region	Resource that is a geographical region
City	Resource that is a city
Facility	Resource that is a facility
Technology	A technology type resource
ContactInformation	Email address, phone number
Position	A position type resource
Patent	A patent type resource
Product	A product type resource
LegalEntity	e.g. stock exchange
Role	Any resource that has a role in something
EventRole	*Any resource that has a role in some business event*
AcquiringCompany	Company to be acquired
CompanyInvestor	Investor of company
ChangingEmployee	Employee to be changed
Distributor	Distributor role in any business event
MergingCompany	Company to be merged
ExpandingCompany	Company to be expanded
EventDescription	*Any resource that elaborates the business event*
ExpansionType	New market or new unit
DividendType	Cash, share
AccountingChangeType	Fiscal year
AffiliationType	Subsidiary, spinoff or division
FilingType	Form 10k, annual report
EmploymentChangeType	Leave type, entry type
Status	Announced, applied, delayed, certified
FinancialTrend	Upgraded, downgraded higher
CompanyMeetingType	Annual meeting, shareholders meeting
ListingChangeType	Leave, enter
FinancialMetric	Metric net loss, metric earnings per share
Event	*A business event resource*
CompanyGrowth	Any resource that represents company growth event
CompanyBasicInformationChange	Any company information change event resource
Acquisition	Company acquisition event resource
CompanyInvestment	Company investment event resource
CompanyExpansion	Company expansion event resource
CompanyReorganization	Company reorganization event resource
CompanyAccounting Change	Company accounting change related event resource
PatentEvent	Any resource that represents a patent event
PatentIssuance	Patent issuance event resource
PatentFiling	Patent filing event resource
LegalIssue	Legal issue related event resource
CompanyReportingEvent	Company reporting related event resource
ProductEvent	Product related event resource
AnalystEvent	e.g. new credit rating of company
WorkEvent	e.g. company layoff event resource
StockEvent	Stock related event resource

The business information classes summarized in Table 1 is the type of data that will be provided by the TM system as such information is discovered from input documents by using TM techniques. *Entity* refers to something that exists by itself such as company, stock exchange or a physical location. *Event* refers to an occurrence that is related to an entity or a set of entities and has some business significance. For instance, a patent filed by a company is of type *PatentEvent* or a change in company's net income is of type *AnalystEvent*. A new product launch is of type *ProductEvent*. *EventDescription* help in explaining further the details of a particular event e.g. the direction of change in net income in an *AnalystEvent* or the year of launch in a *ProductEvent*. Figure 2 shows the hierarchy of business events that are part of CoProE.

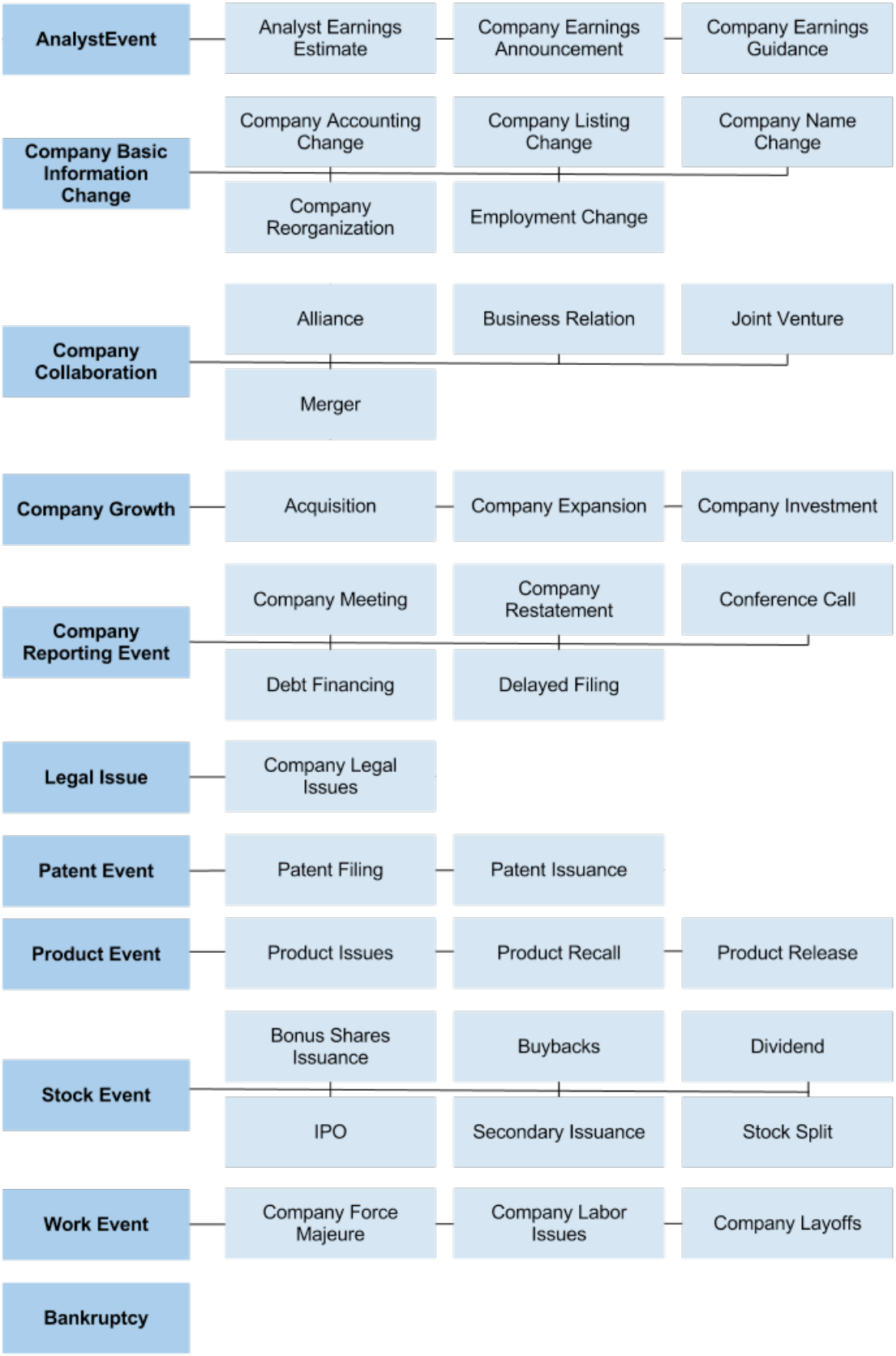

Figure 2 – CoProE ontology events hierarchy

As shown in Figure 2, events are divided into 11 *event categories* that with the exception of Bankruptcy are divided into subcategories, referred to as *event types*.

2.4 Applying CoProE for Collecting and Analyzing Business Intelligence

Ontology is an extremely useful means of storing and representing knowledge, which can further be used to find existing and new knowledge [17, 18, 19]. The extent to which ontology based approaches for BI have developed is limited. *MUlti-industry, Semantic-based next generation business INtelliGence* (MUSING) funded by the European Union is the most notable project for using ontologies and SW technologies for BI TM [20]. MUSING focuses to aid businesses in areas such as financial risk management. There are other systems for BI (*KMP*, *JV-FASTUS*, *Paramendies*) that focus on different domains and capabilities [27]. One of the biggest limitations that most systems have is the lack of an intelligent data system that could store real world business events.

The DAVID system differs from the systems referred above because it is not only extracting specific pieces of information from texts, but also detect complex patterns and relations between them by discovering and extracting respective business events [27]. The objective of using a BI system is to empower its user in making strategic decisions in realistic business productions and operations. There is a large amount of semi-structured and unstructured business events information scattered over the Internet in technology news web sites, web directories etc.

The existing BI systems cannot adapt with the amount of new information emerging in the business domain daily. In a BI system information retrieval is critical function but most BI systems rely on traditional keyword searching for their primary retrieval mechanism [21]. CoProE ontology provides a consistent knowledge base for business events and related information. It combines events collected from different information sources and provides uniform interface for TM system to business information. CoProE assures the consistency, accuracy, invariability, reuse and sharing of the business knowledge.

The DAVID system needs business resources (i.e. companies, products) so it can utilize them for searching relevant events from data sources. The *Information Extraction* (IE) component of the TM system called BEECON [27] implements mapping rules according to existing resource in CoProE. It means that only relevant concepts are searched in the inputs documents. The events form the core of knowledge that the TM system requires. The importance of business events highlights the significance of CoProE as a provider of business resources.

The DAVID system will insert the business information into CoProE through a Java API that is implemented as part of this thesis. In order for the TM system to be able to provide such type of business information it needs CoProE to contain data about companies, their products and competitors. It is this data about companies that we refer in this thesis to as *CoProE background knowledge*. This relationship can be best understood through Figure 3.

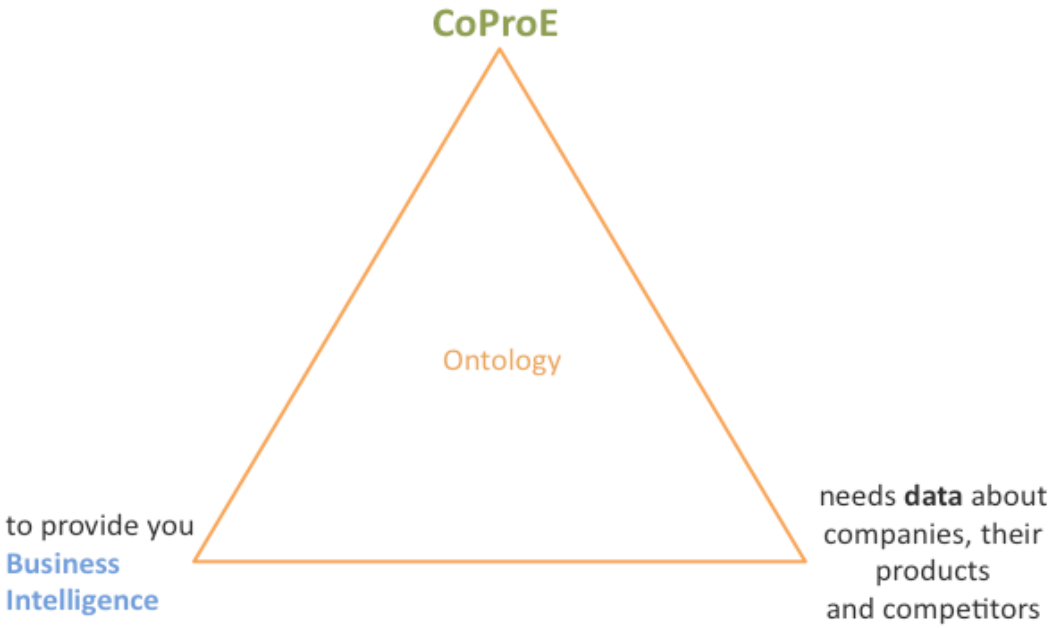

Figure 3 –Relationship of CoProE with business data and BI

As shown in Figure 3, the problem has a triangular relationship. CoProE simply put is a knowledge base for business entities as well as business events stored in a specific format. It needs to store data about entities and events so that they can be retrieved and searched efficiently whenever required by the TM system, which would further process it to provide BI information to the end user. In

order to see where CoProE positions itself in the DAVID system Figure 4 is provided below.

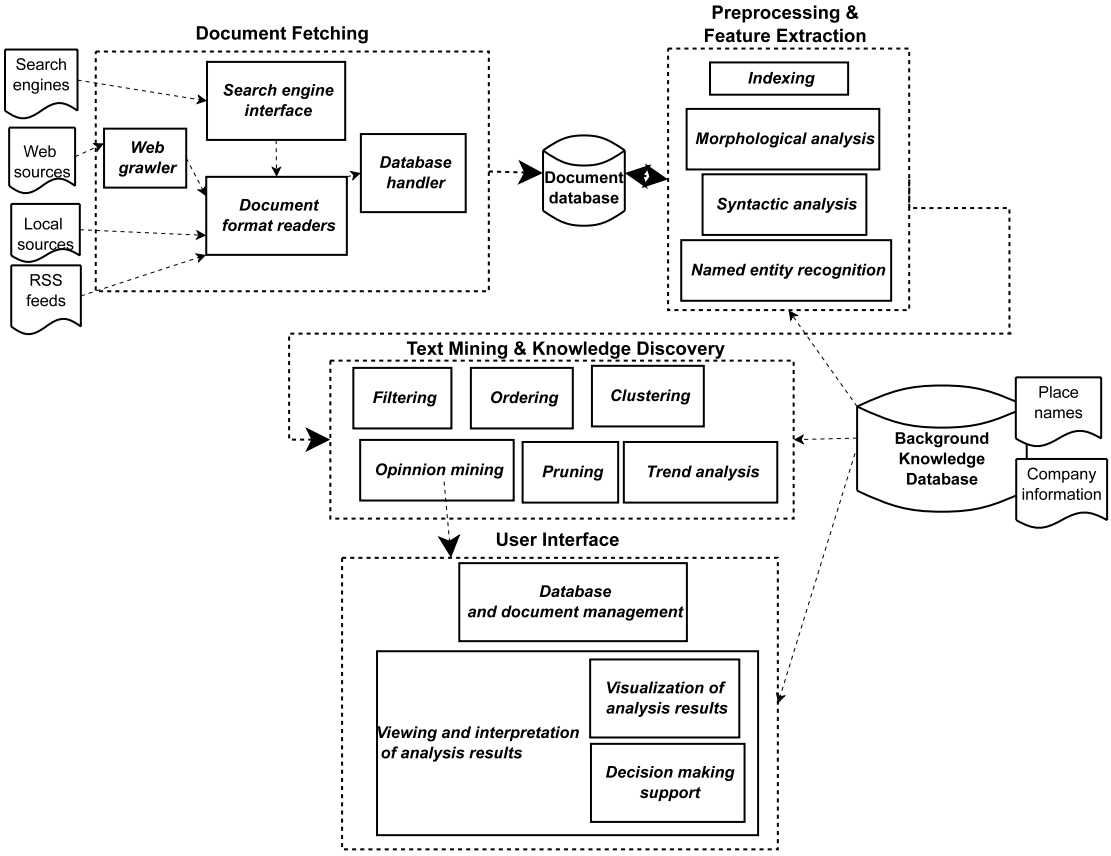

Figure 4 – DAVID system architecture

As illustrated in Figure 4, CoProE is a part of the background knowledge base of DAVID that interfaces with knowledge discovery and text mining components. It is in particular involved in the recognition and extraction of business entities (companies, products etc) and events (mergers, bankruptcies, investments etc). Eventually knowledge is what the user consumes through the system; in that context knowledge base also interfaces with the user interface of the system.

3 Data Collection and Ontology Population

3.1 Categories of Required Data

The DAVID system that uses CoProE is able collect and analyze business events about companies whose data is stored in the ontology. The coverage and accuracy of this process is directly linked with the quality and variety of data available in the ontology. Hence, we wanted to find out which type of business

events and market news would be relevant and interesting for a company. This research led us to define set of categories of data that was to be acquired from participating companies. The categories are listed in the following table.

Table 2 – Categories of required data and sample events within each category

Category of Data	**Sample Events**
Employees Information	• Change of CEO • An important new hire
Financial Information	• Drop in company shares value • Competitor new valuation
Products Information	• Launch of new product • Product recall
Sales Information	• Increase in company profits • Change in competitors sales
Company Geographic Information	• Opening of new office
Supplies Information	• Supplier has complaints about quality control • Political unrest in supplier region
Clients/Customer Information	• Customer feedback about products/services
Legal Information	• Filed new patent • Competitor company filed new patent • Filed lawsuit
Competitors Information	• New product launched by competitor • New technology acquired by competitor

The categories listed in Table 2 served as the starting point from where we built a comprehensive methodology for collecting data. A category in itself is linked with different types of business events as evident from the table above e.g. a

patent related event belong under the category Legal Information, a product related event is placed under the category Products Information.

We analyzed each category in detail and reasoned about the kind of information and granularity required for each category to be able to capture various business events. It was established that information about a company products or services would form the most pivotal information, however, basic information for other categories was deemed equally important. This exercise set the bases on which the data collection questionnaire in Appendix A was constructed. Each section of the questionnaire represents a logical division and all sections together capture the information for the categories mentioned in Table 2.

3.2 Questionnaire

3.2.1 Design

The questionnaire was developed iteratively and we resorted to asking only the essential information to keep the time required from the participants to minimum. It served as the communication point for gathering background knowledge about participating companies. Every question specified in the questionnaire seeks information about the company that could be useful in some kind of BI analysis. Having this goal in mind, the questionnaire was divided into six main sections, each representing a group of business information. Table 3 summarizes the six sections of the questionnaire.

Table 3 –Main sections of the data collection questionnaire

Questionnaire Sections	**Type of Information Sought**
Company Profile	Company name, geographic location, target market, key personnel, financial growth
Products and Services	Products or services of the company
Legal Information	Company patents, trademarks, legal disputes
Competitors Information	Competitors' products, services and legal information
Clients/Customers Information	Clients' or customers' profiles, their feedback/reviews on products and services
Miscellaneous Information	Any other linked companies e.g. suppliers, contractors

As shown in Table 3, the first section of the questionnaire deals with company profiles. The company name can be used to track mention about it on any kind of digital media such as newspaper, blog, social network or financial magazine. It is useful to know about industry a company belongs to as that gives a general idea of what type of companies would be its competitors and the kind of market news that would be relevant for the company. Information about company geographical locations and main market areas is important as it reveals if the company has a global reach or local competition. If the company is subsidiary to a larger parent company then any business event related to the parent company can potentially affect it or vice versa. Financial data helps to evaluate the growth of a company or track stock market events.

The most important section of the questionnaire is the 'products and services' section. They both are synonymous in the sense that that some companies are focused on products while others are on services and some might be focusing equally on providing both. The information required for both of these categories is similar. Knowing about product or service is important so that any type of event where they are mentioned could be captured. It could be, for instance, a release of a competing product, customer opinion about a certain product or

service, discount deal on a certain product by another company, cheaper alternatives in the market for a particular product category, discussion about a service on a blog or social network, market news about a product.

In the questionnaire, we requested for all possible information sources through which we could acquire this crucial information. A company might have many products in multiple categories and might not be interested or have the time and resources to provide all the information we required. Hence, we wanted the companies to provide us information at the very least about the products that keep them sustainable and provide competitive advantage. If we would not be able to collect and analyze news concerning all their products and services, we could provide information about the most important once. This should make the software system based on CoProE valuable to the company in question.

The 'legal information' section primarily deals with patent information. Patents help gauge the level of research and innovation in a company. The type of patents a company owns tells the kind of patent events that would be of interest to the company. A patent is essentially an investment by a company. Good knowledge about patents can be used to notify about other invention activities occurring within the particular patent area that a company could stay informed of. For instance, if a company *X* has registered for patent in a new type of technology that tells which area company *X* is exploring for innovation.

The section on competitor information is also one of the most valuable ones from the perceptive of the DAVID system and the analysis it can provide. Good competitor intelligence can only be gathered if one has real information about the competitors. Hence, knowing the names of competitor companies and their products and services is vital to the DAVID system, so that similar type of product and patent events as well as news stories could be detected. For instance in case a competitor company registers for a new patent, one has a clear idea of what innovation strategy they are adapting to and what novelty could be expected in their future products. If we find people discussing and appreciating a competitor product it's a clear indication of what people like and are talking about.

The next section, 'clients and customer information', primarily seeks to gain information about reviews and feedback sources about products or services of the company in question. Key clients information could be used by the system to search for similar type of clients that would be interested in a product or service that the company offers. The sources of feedback are from where customer opinion and sentiments could potentially be drawn out e.g. a blog where a student talks about experience with a course. It would be ideal for the system if the company has feedback or reviews stored in digital form so it could access those directly.

The final section aims at colleting information about companies, suppliers or contractors that are in any way linked to the company. A business event or news about any entity that is linked to a company is indirectly of interest to the company or depending upon the relationship with the company could potentially affect it. For example, if a political upheaval in region where the company's supplier is based is significant news for the company. Finally, we were interested to know if we missed any point that the company would like to tell us that is important from a BI point of view.

3.2.2 Implementing the Survey

All of the seven companies that participate in the 'Towards e-leadership' project was sent the questionnaire through email. We requested them to fill it in completely and respond within two weeks. For the assistance of the companies we also provided a filled in sample questionnaire. The companies were explicitly informed that all shared information is under a non-disclosure agreement, will be kept highly confidential and stored in a secure manner. The companies were told that they could leave any sections of the questionnaire blank, if they felt a certain piece of information could not be disclosed.

The success of the data collection exercise was dependent on the type and detail of response we receive from the participating companies. Hence, it was important to get as much data from the companies as possible. The population of ontology is, naturally, directly affected by the amount of data we were able to gather about each company.

In total five companies responded to the data collection survey out of which three companies, (namely Company A, Company C & Company D) responded to our questionnaire. Out of these three companies two companies, A and D, sent us catalogs about their products or services. As for rest of the companies, they provided basic profile about the company and redirected us to their website for gathering more information. We were specifically interested in information about competitors and legal matters, such as patents. However, only Company A provided some competitor information and no one provided any legal information. The competitor and legal information could provide an interesting test data for tracking a company as well as its competitor's market activity. We also realized that no company had a digital data store for customer feedback and reviews that we could utilize for analyzing customer sentiments. Or if they had such a system, they were not willing to provide access to it.

We decided to perform an extensive online research about each company and populate the ontology with as much additional information as we could gather from public sources, such as the company website and relevant news articles. We were able to accumulate significant products and services information for the same companies who responded to the questionnaire as well as for one more company that did not respond to the questionnaire.

3.3 Ontology Population

The DAVID system needs access to the gathered data in order to utilize it for information collection and processing. The second and final stage of the background knowledge collection phase involved storing the gathered data in the CoProE ontology in an organized manner. The ontology has the flexibility to store, classify and establish relationships between various kinds of data. We created a separate ontology file for each company that only stores data relevant to that particular company.

Protégé [22] is a free, open-source platform that provides numerous tools to construct domain models and knowledge-based applications with ontologies. It was utilized to populate CoProE with business data. Figure 5 shows portion of

Protégé user interface and how a CoProE ontology company resource appears after its properties have been inserted and linked to it.

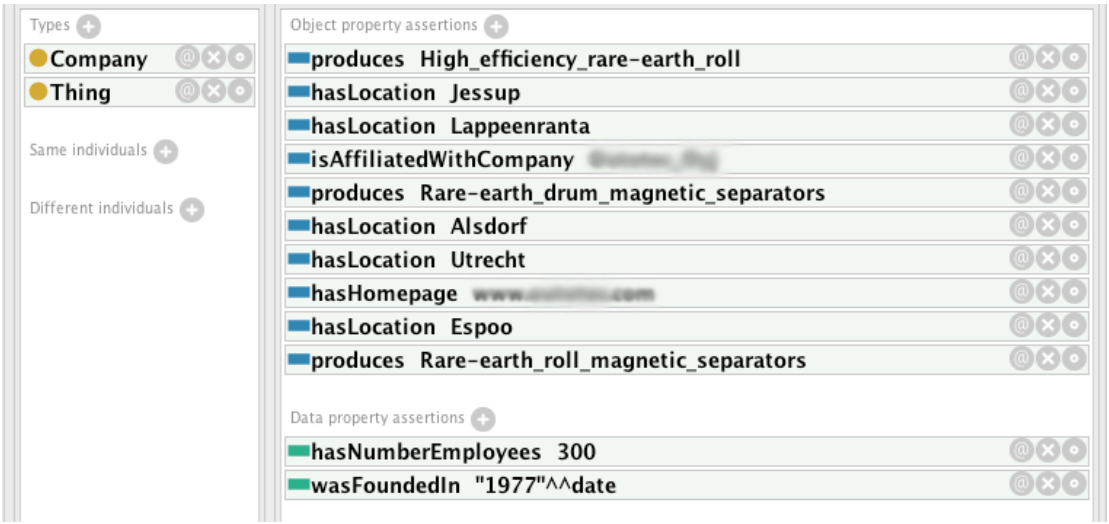

Figure 5 - A populated company resource in Protégé. Object property assertions are the links the company has to the other ontology *resources* and data property assertions are the links of company to data-type values or literals.

The population starts by defining the company as an ontology resource and then setting company's various object and data properties such as location(s), website, number of employees, year founded etc. Every product of a company is a separate ontology resource that was linked with the relevant company using primarily *isProducedBy* predicate. The following table demonstrates the number of predicates that we were able to establish for each company.

Table 4- Number of predicates for each company

Company	**No. of *isProducedBy* relations**	**No. of *isAffiliatedWithCompany* relations**
Company A	14	-
Company B	-	47
Company C	67	-
Company D	19	-
Company E	4	-

For the confidentiality reasons, the real names of the companies have been replaced by aliases in Table 4 and throughout the whole thesis. According to the information collected, Company A produces 14 products, Company C produces 67, Company D 19 and Company E produces 4 products. Company B has affiliation with 47 products, which means that it is not necessarily directly involved in product creation, e.g. it owns a property where certain service is offered.

Products form the main source of business information that the ontology consists of. As we mentioned above, the ontology is based on two constructs: newsEvents ontology and UNSPSC. Table 5 provides some examples of products and their UNSPSC classification.

Table 5- Products or Services UNSPSC classification

Product Name	**UNSPSC Classification**
Company A Tractor Series	Commercial and Military and Private Vehicles and their Accessories and Components/Motor Vehicles/Specialized and recreational vehicles/Agricultural tractors
Spiral Concentrator of Company C	Industrial Manufacturing and Processing Machinery and Accessories/Raw materials processing machinery
Educational Online Forum of Company D	Education and Training Services/Vocational training/In service training and man power development
Company E Software Product	Communications and Computer Equipment and Peripherals and Components and Supplies/Software

As shown in Table 5, every product resource in the ontology is classified under the UNSPSC classification of products and industries. Let's consider Company A Tractor Series as an example. It is evident from the table that it is an agriculture tractor and all agriculture tractors are considered motor vehicles as well as

specialized and recreational vehicles. On the other hand, Company E Software Product is a software product subsumed under a broad classification of Computer Equipment and Peripherals.

For two of the companies, we populated the ontology with similar information about their competitors. Competitor company relationships are achieved by linking the two companies using *isCompetitorOf* predicate in the ontology. The competitor's information will serve as trial information for the DAVID system in providing BI to a company about their competitors.

4 Implementation of Ontology API

This section elaborates the design and implementation details of the *CoProE API* that was developed in order to programmatically insert and access business events from the CoProE ontology. The API provides an interface for the DAVID system to manipulate the ontology and add new facts into it. The API is composed of different components. These components are explained in the following sections.

4.1 Development Tools and Language

Java programming language [23] and Eclipse *Software Development Kit* (SDK) [24] has been used as development environment of the API. Java is one of the most popular languages in the world and it is used in various software disciplines. It is estimated that 1.1 billon desktops run Java. Java powers printers, webcams, games, car navigation systems, medical devices, parking payment station and more. One of the strongest qualities of Java is of platform independence, implying that you can write software on any platform and run it virtually on any other platform.

Eclipse SDK is a cross-platform powerful tool that has strong combination with Java and reduces the time and cost spent on writing code. The Eclipse SDK consists of the Eclipse Platform, Java development tools and a Plug-in development environment. The Eclipse platform is a multi-language software development environment consisting of an *Integrated Development Environment* (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used

to develop applications in various programming language including Java along others such as C++, Python, Ruby, PHP, Perl.

The CoProE API utilizes a third party Java framework called Jena [25] to communicate with CoProE ontology. Jena is an extensive open-source Java based framework that provides various programmatic tools to build SW applications. It Apache Software Foundation actively supports Jena. Researchers in HP Labs originally developed Jena in year 2000 and since then it has been extensively used in a wide variety of semantic web applications and demonstrations [26]. It is a well-known framework in the SW community with good documentation and support available. Jena provides interfaces and methods to load ontology models into memory and manipulate them. A model can be based on data from files, databases, URLs or a merger of all. The framework provides detailed class and method documentation generated from Jena source code. It can also be extended through external components to provide powerful query engine.

There are many other SW frameworks available, but the DAVID system is being developed in Java and hence Jena made a good choice for API development, as it would not lead to any integration issues. The sections that follow will discuss in more detail how the API utilizes the Jena framework.

4.2 Architecture of API

The API consists of three components: *Ontology Interfacer* (OI), the *Event class* and various objects that are sub-classed from the Event class. These three components are dependent on each other and perform tasks in a linear manner. The placement of these components in the API, their composition, and internal plus external workings is illustrated in Figure 6.

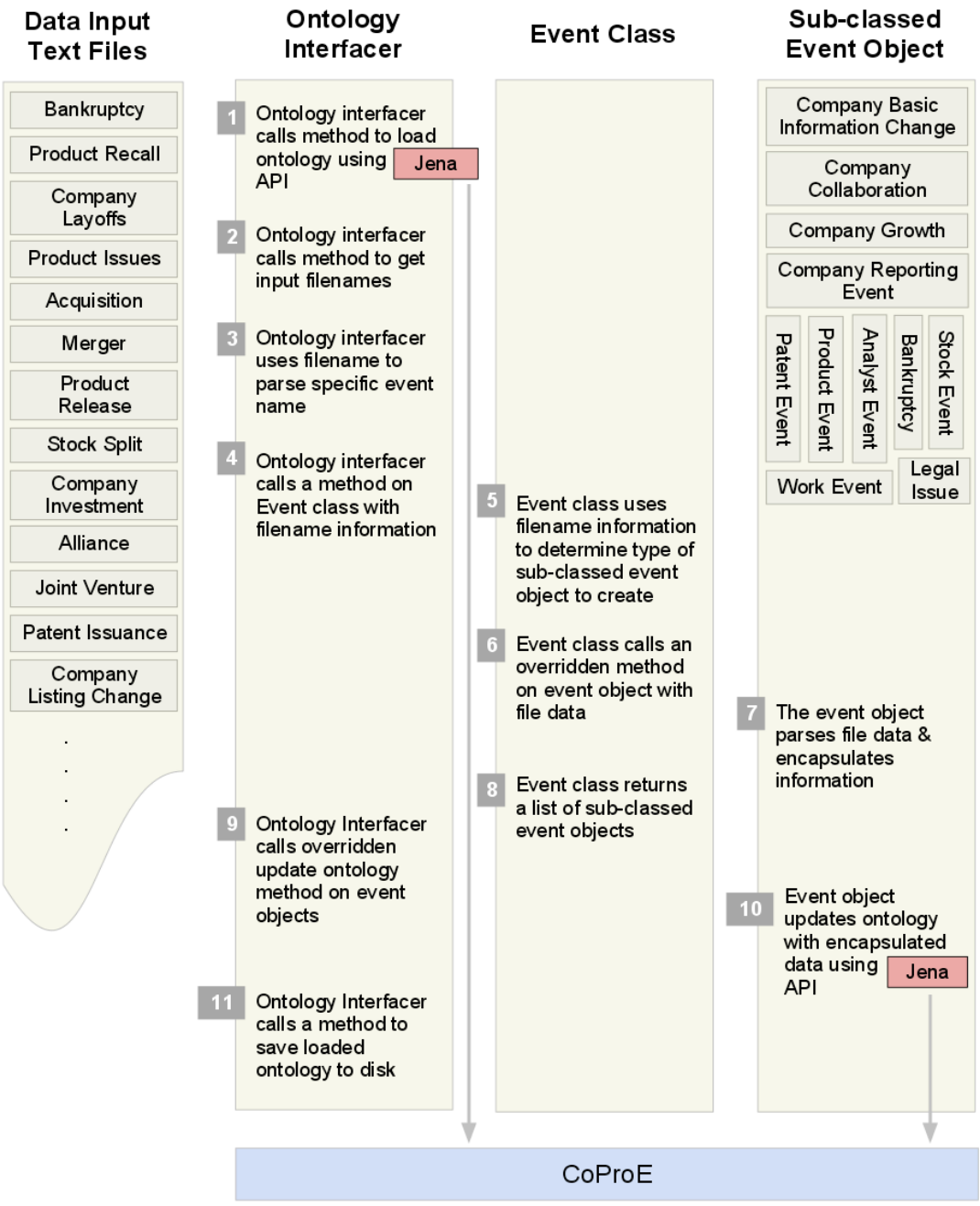

Figure 6 – Structure and communication flow of CoProE API

The first brown rectangular shaped block in Figure 6 represents the collection of text files containing events data that is to be inserted into the ontology. It is technically not part of the API, but only an input source for it. The other three brown rectangular blocks represent components of the API. OI initiates all the method calls to different components of the API. All the *sub-classed event objects* (SCEO) must implement the abstract class *Event* to become functional component of the API. The last rectangular block, at any given point in time,

represents exactly one SCEO that can be from any of the assortment shown in upper part of the last rectangular block. The numbered steps in the figure show the communication flow between the different components and explain how data flows from text files into the ontology. Every numbered step inside the blocks is a Java method belonging to the API, some of whose pseudo code has been provided in the section (Sections 4.4, 4.5) discussing the respective component. The API interfaces with Jena, which in turn communicates with the CoProE ontology, at two points indicated by the red boxes in Figure 6.

Object-oriented principles of *inheritance* and *polymorphism* have been extensively used in developing CoProE API. In object-oriented programming inheritance is a way to reuse code of existing objects, establish a subtype from an existing object, or both. Polymorphism allows values of different data types to be handled using a uniform interface. Step number 6 and 9 in Figure 6 show the points where polymorphism is used.

4.3 Data Input Text Files

IE and event extraction component called BEECON is developed as a part of the effort to build the DAVID system [27]. BEECON provides the input to the API and serves as the communication channel between CoProE API and the TM system. Our initial understanding was that the communication mechanism between the BEECON component and API would be XML-based, but on further analysis we decided that XML does not provide us any additional benefit as the data is only used internally by our TM system. In fact, formatting the IE output to XML and extracting information from it would lead to additional overhead and would thus hamper the efficiency of the whole system. Hence, plain text files were opted for as the most efficient method of data exchange between the BEECON component and CoProE API.

BEECON automatically extracts business entities and events from text sources collected by the DAVID system. It provides this information as input to CoProE API. The input text files not only provide data related to a business event but the filenames are also used to determine the type of SCEOs to instantiate. The sections that follow explain the naming format, structure and internal content of

the different input files that the API supports. We also discuss the meaning of data contained in the text files.

Analyst Event Category

Text files with the name format '*AnalystEvent_xxx.txt*' provide Analyst Events where *xxx* is the name of any of the specific analyst events. The three possible analyst event types are listed in Table 6.

Table 6 – Analyst events input file example. The original sentences from which the triplets were sourced are denoted in italics. The brown background in the table represents portion that is directly coming from text file and each line with brown background represents a single row of the text file.

Analyst Earnings Estimate Event
Separately, analysts at Deutsche Bank (NYSE: DB) raised their price target on shares of DuPont to $58.00 in a research note to investors on Monday, June 6th.
[AnalystEarningsEstimate, hasAnalyst, Deutsche Bank]
[AnalystEarningsEstimate, hasMoney, $58.00]
[AnalystEarningsEstimate, usesFinancialMetric, shares]
[AnalystEarningsEstimate, hasAnalysedCompany, DuPont]
Analysts surveyed by FactSet Research had produced a consensus forecast of 74 cents a share.
[AnalystEarningsEstimate, usesFinancialMetric, a share]
[AnalystEarningsEstimate, hasMoney, 74 cents]
[AnalystEarningsEstimate, hasParticipant, FactSet]

Table – 6 continues

Company Earnings Announcement Event
Honda reported a 90 percent fall in quarterly operating profit on Monday, versus expectations of a loss, after it suffered the biggest production drop by any car maker from the March disaster, due mainly to bad timing for the scheduled delivery of parts.
[CompanyEarningsAnnouncement, hasAnnoucingCompany, Honda]
[CompanyEarningsAnnouncement, usesFinancialMetric, quarterly operating profit]
[CompanyEarningsAnnouncement, hasFinancialTrend, fall]
[CompanyEarningsAnnouncement, hasPercent, 90 percent]
Russian crude producer, Rosneft, has posted a 2Q 2011 net income of $2.831 billion under US GAAP.
[CompanyEarningsAnnouncement, hasAnnoucingCompany, Rosneft]
[CompanyEarningsAnnouncement, usesFinancialMetric, 2Q 2011 net income]
[CompanyEarningsAnnouncement, hasMoney, $2.831 billion]
Company Earnings Guidance Event
Prior guidance was a range of $3.65 to $3.85 per share for DuPont, excluding the impact of Danisco.
[CompanyEarningsGuidance, hasAnnoucingCompany, DuPont]
[CompanyEarningsGuidance, usesFinancialMetric, per share]
[CompanyEarningsGuidance, hasMoney, $3.65]
AEP reaffirmed its ongoing guidance range for 2011 of between $3.00 and $3.20 per share.
[CompanyEarningsGuidance, hasAnnoucingCompany, AEP]
[CompanyEarningsGuidance, usesFinancialMetric, per share]
[CompanyEarningsGuidance, hasMoney, $3.00]

The table above outlines how specific analyst events business data is provided in text files in the form of RDF triplets. Every line consists of an RDF triplet of the

form: [subject, predicate, object]. The subject and object of the triplet are *resources* for the ontology where as the predicate specifies the relationship between the subject and object. One input file can consist of multiple events each separated by an empty line.

For instance, the 'analyst earnings estimate event' sections tells that according to Deutsche Bank a company called DuPont has a share value of $58.00 and FactSet has a share value of 74 cents. According to the 'company earnings announcement' section Honda has announced a fall of 90% in their quarterly profits where as Rosneft had a net income of $2.831 billion in the second quarter of 2011. In company earnings guidance event DuPont announced its share value as $3.65 and AEP announced its share value as $3.00.

It is also pertinent to discuss the structure of the Java class that is used in the API for storing analyst events such as the ones introduced in Table 6. All the information above is encapsulated in a single Java class that is shown in Figure 7.

```
AnalystEvent extends Event
{
    long id
    String companyName
    String financialMetric
    String financiallTrend
    String analysedCompany
    String participant
    Money money
    String percent
    String quarter
    String year
}
```

Figure 7 – Analyst Event Java Class

All the attributes of the Java class in Figure 7 are derived from the information that is specified in Table 6 in the form of RDF triplets. The orange color indicates

the attributes that are inherited from the super class *Event*. The details of the *Event* class are provided in Section 4.5. The two attributes *id* and *companyName* are attributes that are found across nearly all the event types.

The CoProE API populates all the above attributes with information read from the input text files. They all are rather simple string attributes with the exception of the variable *Money* that is a custom class of the API. The three analyst events discussed above could have had three separate Java classes yet only one has been implemented in the API encapsulating any type of Analyst Event. It is done so as across events some attributes are common and having a separate class for each specific analyst event does not provide any benefit. This is a principle that has been followed throughout the development of the API for all parent events.

Bankruptcy Event Category

Text files with the name format '*Bankruptcy.txt*' provide information about Bankruptcy events. Bankruptcy event category does not have any subclasses as shown in Table 7.

Table 7 – Bankruptcy event input file format

Swedish home electronics chain Albertsons LLC filed for bankruptcy on Monday, citing stiff competition, price pressure, and low profits.
[Bancruptcy, hasBancruptCompany, Albertsons LLC]
Nexgen, a chain of consumer electronics stores that used to operate across the U.S., filed for Chapter 11 bankruptcy in January of this year.
[Bancruptcy, hasBancruptCompany, Nexgen]
Though MGM is one of the biggest and best known entertainment companies to file for bankruptcy, it's certainly not the first.
[Bancruptcy, hasBancruptCompany, MGM]

As one can observe from Table 7, three companies, Albertson LLC, Nexgen and MGM, have declared bankruptcy and that is only the information that is to be encapsulated in the Bankrucpty class shown below.

```
Bankruptcy extends Event
{
    long id
    String companyName
}
```

Figure 8 – Bankruptcy Event Java Class

As Figure 8 shows, Bankrupty event only requires two attributes that it directly inherits from the Event class.

Company Basic Information Change Event Category

Company Basic Information Change events originate from text files with the name format '*CompanyBasicInformationChange_xxx.txt*' where *xxx* is the name of any of the five company basic information change event types shown in Table 8.

Table 8 – Company Basic Information Change events input file format

Company Accounting Change Event
Verizon will take a $600 million charge in fiscal 2010 for changes made to its method of accounting, a move that comes on the heels of rival AT&T's similar announcement last week.
[CompanyAccountingChange, hasAccountingChangingCompany, Verizon]
AT&T expects the impact of this accounting change on its fourth-quarter 2010 results to be a pretax, noncash charge of approximately $2.7 billion, or $0.28 per share.
[CompanyAccountingChange, hasAccountingChangingCompany, AT&T]

Table – 8 continues

Company Listing Change Event
Invesco PowerShares Capital Management LLC, a leading global provider of exchange-traded funds (ETFs), announced today that the ticker symbol for PowerShares QQQ will be changed from "QQQQ" to "QQQ," effective March 23, 2011.
[CompanyListingChange, hasListingChangingCompany, Invesco PowerShares Capital Management LLC]
[CompanyListingChange, hasNewListing, QQQ]
Atlas Energy, L.P. announces ticker symbol change to ATLS.
[CompanyListingChange, hasListingChangingCompany, Atlas Energy, L.P.]
[CompanyListingChange, hasNewListing, ATLS]
Company Name Change Event
The Knot, Inc. (NASDAQ: KNOT), the premier media company devoted to weddings, nesting and babies, today announced that it had received stockholder approval to change the company's name to XO Group Inc.
[CompanyNameChange, hasOldCompanyName, The Knot, Inc.]
[CompanyNameChange, hasNewCompanyName, XO Group Inc.]
Members are hereby informed that the name of Swaraj Mazda Limited shall be changed to SML Isuzu Limited.
[CompanyNameChange, hasOldCompanyName, Swaraj Mazda Limited]
[CompanyNameChange, hasNewCompanyName, SML Isuzu Limited]
Company Reorganization Event
Cisco's company reorganization met with employee optimism.
[CompanyReorganization, hasReorganizedCompany, Cisco]
AOL Execs Depart Amid Another Company Reorganization.
[CompanyReorganization, hasReorganizedCompany, AOL]

Table – 8 continues

Employment Change event
Nick White joined Talend as Chief Financial Officer (CFO) to take on responsibility for all of Talendís global finance and administration operations.
[EmploymentChange, hasManagementChangingCompany, Talend]
[EmploymentChange, affectsPosition, Chief Financial Officer]
[EmploymentChange, hasParticipant, Nick White]
and Scott Devens joined Talend as Vice President and General Manager of the Application Integration division.
[EmploymentChange, hasManagementChangingCompany, Talend]
[EmploymentChange, affectsPosition, Vice President]
[EmploymentChange, hasParticipant, Scott Devens]

The table above outlines how different 'company basic information change' events data are provided in text files in the form of RDF triplets. We come to know that Verizon & AT&T had a change in the company that deals with their accounting. Invesco PowerShares Capital Management LLC and Atlas Energy have a new listing on the stock market where as The Knot and XO Group both have a new company name. Cisco & AOL went through company reorganization and Talend Chief Financial Officer and Vice President were changed. It might appear from the table that there is extensive information that needs to be captured. However, all possible information from different types of events above can be encapsulated in a single Java class shown in Figure 9.

```
CompanyBasicInformationChange extends Event
{
      String newListing
      String newCompanyName
      String participant
      String affectedPosition
}
```

Figure 9 – Company Basic Information Change Event Java Class. The two attributes (id and companyName) inherited from the super class *Event* are omitted[1].

As one can see in the figure above there are four non-inherited attributes that are required depending upon the type of specific Company Basic Information Change event that is read from an input text file. Like other parent events, a single Java class is sufficient to encapsulate data coming from all types of Company Basic Information Change events.

Company Collaboration Event Category

Company Collaboration events come from text files with name format '*CompanyCollaboration_xxx.txt*' where *xxx* is the name of any of the specific company collaboration event types. Specific company collaboration events are of the four types as shown in Table 9.

[1] The inherited attributes *id* and *companyName* are omitted from the rest of the event subclass examples (Figures 9-14).

Table 9 – Company Collaboration events input file format

Alliance Event
Nokia and Microsoft enter strategic alliance on Windows Phone, Bing, Xbox Live and more.
[Alliance, hasAllyingCompany, Nokia]
[Alliance, hasAllyingCompany, Microsoft]
Based on the strategic alliance between NEC and Lenovo, NEC Lenovo Japan Group comprises of three companies: the holding company, Lenovo NEC Holdings B.V. and NEC Personal Computers Ltd.
[Alliance, hasAllyingCompany, NEC]
[Alliance, hasAllyingCompany, Lenovo]
Business Relation Event
Elan and PPD form global business relationship.
[BusinessRelation, hasRelatedCompany, Elan]
[BusinessRelation, hasRelatedCompany, PPD]
Joint Venture Event
Daimler AG and Bosch GmbH have formally established a 50:50 joint venture (JV) for electric motors.
[JointVenture, hasVenturingCompany, Daimler AG]
[JointVenture, hasVenturingCompany, Bosch GmbH]
NEC and Lenovo both hold stakes in the joint venture.
[JointVenture, hasVenturingCompany, NEC]
[JointVenture, hasVenturingCompany, Lenovo]
Merger Event
Deutsche Boerse Co. and NYSE Euronext to merge.
[Merger, hasMergingCompany, Deutsche Boerse Co.]
[Merger, hasMergingCompany, NYSE Euronext]

It is evident from table above that this category of events is about cooperation and alliance between companies. It is reporting an alliance between Nokia and Microsoft as well as NEC and Lenovo. Furthermore, they have also started a joint venture and so have Daimler AG and Bosch. Elan and PPD are related companies; this is more of business knowledge then a business event. Companies that have merged into each other are Deutsche Boerse Co., NYSE Euronext.

As can be observed in all the event examples above, the information that needs to be stored about companies come as pairs i.e. two merging companies or two allying companies. Hence, the Java class in Figure 10 is also designed to read this kind of information easier.

```
CompanyCollaboration extends Event
{
    ArrayList<String> collaboratingCompanies
}
```

Figure 10 – Company Collaboration Event Java Class

The ArrayList of Strings in Figure 10 serves the purpose of reading a pair of companies with similar relationship and in this case the inherited attributes are unused but to retain compatibility the class also inherits from Event class.

Company Growth Event Category

Company Growth events come from text files with the name format '*CompanyGrowth_xxx.txt*' where *xxx* is the name of any of the specific Company Growth event types. Specific Company Growth events are of three types as shown in Table 10.

Table 10 – Company Growth events input file format

Acquisition Event
AT&T's $39 billion plan to acquire Deutsche Telekom's T-Mobile unit got support from a Louisiana regulator and 11 state attorneys general on Wednesday.
[Acquisition, hasAcquiringCompany, AT&T]
[Acquisition, hasAcquiredCompany, T-Mobile]
Six months after acquiring European movie-rental site Lovefilm International Ltd., Amazon Co. recently agree to buy The Book Depository Inc., U.K.'s largest dedicated online bookseller that provides free worldwide delivery of over six million book titles.
[Acquisition, hasAcquiringCompany, Amazon Co.]
[Acquisition, hasAcquiredCompany, Book Depository Inc.]
Company Expansion Event
The expansion is a boost to the stock of the fast-growth video service provider because the move signals Netflix's first expansion to the international market beyond the United States and Canada, where it currently operates.
[CompanyExpansion, hasExpandingCompany, Netflix]
Hawaiian Airlines continues its rapid expansion in Asia with the much-anticipated launch of its direct Osaka service next Tuesday a trip being touted with as much fanfare as the launch of direct Seoul service earlier this year.
[CompanyExpansion, hasExpandingCompany, Hawaiian Airlines]
Company Investment Event
Kohlberg Kravis Roberts Corp has agreed to invest $113.8 million in convertible bonds of Singapore-listed Chinese water treatment company United Envirotech (UEL), as the U.S. private equity fund seeks to tap the mainland's fast-growing water treatment industry.
[CompanyInvestment, hasInvestor, Kravis Roberts Corp]
[CompanyInvestment, hasInvestmentObject, United Envirotech]
[CompanyInvestment, hasMoney, $113.8 million]

All the business events in this category are related to growth of a company. As you can see in Table 10, AT&T has acquired T-Mobile where as Amazon has acquired Book Depository. Netflix and Hawaiian Airlines had a company expansion whose details are unspecified. United Envirotech invested $113.8 million in Kravis Roberts Corp. Most of the information consists of straightforward string attributes, with the exception of the Company Investment event, which requires money attribute. Hence, the Java class in Figure 11 suits all the requirements of this event type well.

```
CompanyGrowth extends Event
{
    Money money
    String investmentObject
    String acquiredCompany
}
```

Figure 11 – Company Growth Event Java Class

The inherited attribute companyName in Figure 11 is used to store acquiring company where as acquiredCompany attribute complements the relationship. Money attributed is populated wherever encountered during reading from text file.

Legal Issues Event Category

Legal Issues events come from text files with the name format '*LegalIssue_CompanyLegalIssues.txt*'. There is only one type of event belonging to legal issues category as shown in Table 11.

Table 11 – Legal Issue events input file format

Company Legal Issues Event
Samsung is no longer in these talks after the increase in legal issues between Apple and Samsung.
[CompanyLegalIssues, hasPlaintiff, Apple]
[CompanyLegalIssues, hasSuedEntity, Samsung]
HTC infringe Apple patents; legal problems will now hit all Android manufacturers.
[CompanyLegalIssues, hasPlaintiff, Apple]
[CompanyLegalIssues, hasSuedEntity, HTC Co.]

As you can see in Table 11 the event has pair of complementing attribute, in first event Apple has sued Samsung and in second event Apple has also sued HTC. The only information that needs to be stored in this event is plaintiff name and the company being sued therefore these two attributes are present in the Java class in Figure 12.

```
class LegalIssue extends Event
{
      String plaintiff
      String suedEntity
}
```

Figure 12 – Legal Issue Event Java Class

In Figure 12 the inherited attribute companyName is useless for company legal issues event but for compatibility purpose the class is inheriting from Event. Its also possible that in future the event is extended in some other way and companyName attribute could be used.

Patent Event Category

Patent events come from text files with the name format '*PatentEvent_xxx.txt*' where *xxx* is the name of any of the specific Patent event types. Specific Patent events are of two types as shown in Table 12.

Table 12 – Patent events input file format

Patent Filing Event
The first pictures of the 2012 Subaru Impreza XV have leaked to web following a patent filing.
[PatentFiling, hasPatentHoldingCompany, Subaru Co.]
Apple patent filing reveals plans for transparent augmented reality iPad display technology.
[PatentFiling, hasPatentHoldingCompany, Apple]
Patent Issuance Event
Proteus Co. Announces Issuance of U.S. Patent for Ingestible Digital Devices.
[PatentIssuance, hasPatentHoldingCompany, Proteus Co.]
Arch Biopartners Inc. Announces Issuance of New U.S. Patent for Biofilm Inhibitor.
[PatentIssuance, hasPatentHoldingCompany, Arch Biopartners Inc.]

As can be seen in Table 12, Patent event consists of straightforward RDF triplets. Subaru Co and Apple applied for a patent and a patent was issued to Proteus Co. and Arch Biopartners Inc. As you can see in Figure 13 no non-inherited attributes are required to store patent events information.

```
class PatentEvent extends Event
{
}
```

Figure 13 – Patent Event Java Class

Patent event and Bankruptcy event are one of those simplest cases of events, which do not need any data member added to its super class.

Product Event Category

Product events come from text files with the name format '*ProductEvent_xxx.txt*' where *xxx* is the name of any of the specific product events. Specific Product events are of three types as shown in Table 13.

Table 13 – Product events input file format

Product Issues Event
The "Gizmodo" site (http://www.gizmodo.com) says that Apple has released a statement to them, admitting the product issues with iMacs.
[ProductIssues, hasProduct, iMacs]
[ProductIssues, hasProvider, Apple]
Product Recall Event
Konica Minolta recently issued a voluntary product recall for certain laser printers and MFPs sold in Australia between June 2010 and April of this year.
[ProductRecall, hasProduct, MFPs]
[ProductRecall, hasProvider, Konica Minolta]
Konica Minolta recently issued a voluntary product recall for certain laser printers and MFPs sold in Australia between June 2010 and April of this year.
[ProductRecall, hasProduct, laser printer]
[ProductRecall, hasProvider, Konica Minolta]

Table – 13 continues

Product Release Event
HTC Corporation, a global leader in mobile innovation and design, today unveiled three new versions of its most popular and advanced smart phones HTC Desire S, HTC Wildfire S and HTC Incredible S
[ProductRelease, hasProduct, Wildfire S]
[ProductRelease, hasProvider, HTC Corporation]
Micro Irrigation Systems, is introducing a new 60 Hp tractor model 5060E in India.
[ProductRelease, hasProduct, Hp tractor model]
[ProductRelease, hasProvider, Micro Irrigation Systems]

From Table 13 we deduce that iMac produced by Apple has had some issues and Konica Minolta has recalled two of its products: MFPs and laser printer. On other side HTC Corporation has launched a new product by name of Wildfire S and Micro Irrigation Systems has also launched a new Hp tractor model. Only one data member is added to event class to create the Java class in Figure 14.

```
ProductEvent extends Event
{
    ArrayList<String> products
}
```

Figure 14 – Product Event Java Class

For all types of product event the information that needs to be stored is the company name and products that it has released, recalled or has had an issues in. Therefore, as Figure 14 shows, only adding an array for storing information about these products makes the Java class complete to be able to encapsulate data read from any type of product event.

Stock Event Category

Stock events come from text files with name format *StockEvent_xxx.txt* where xxx can be name of any of the specific stock events. Specific stock events are of six types as shown in Table 14.

Table 14 – Stock events input file format

Bonus Shares Issuance Event
Nitin Fire Protection Industries Ltd soared 16% at Rs 126 to its 52-week high on Thursday after the company said it is in talks with the board for a possible bonus issue.
[BonusSharesIssuance, hasSharesIssuingCompany, Nitin Fire Protection Industries]
India Nippon Electricals Inc. closed higher by 9.76% at Rs 277.96 after the company announced bonus issue plans.
[BonusSharesIssuance, hasSharesIssuingCompany, Nippon Electricals Inc.]
Buybacks Event
A year ago, IBM also approved a $15 billion stock buyback program.
[Buybacks, hasSharesIssuingCompany, IBM]
The J.P. Morgan CEO said on a conference call this morning that the giant bank may ask regulators to dole out more of its money in dividends and/or stock buybacks.
[Buybacks, hasSharesIssuingCompany, J.P. Morgan]
Dividend Event
This morning, Monsanto Co. declared its quarterly dividend of 30 cents per share, an increase of about 7% over its prior dividend.
[Dividend, hasMoney, 30 cents]
[Dividend, hasDividendPayingCompany, Monsanto Co.]
According to Dividend Channel, on 8/3/11, Ameriprise Financial Inc (AMP) will trade ex-dividend, for its quarterly dividend of $0.23.
[Dividend, hasMoney, $0.23]
[Dividend, hasDividendPayingCompany, Ameriprise Financial]

Table – 14 continues

IPO Event
A GM IPO could be the largest in U.S. history. It would have to bring in US$70 billion to pay back all of GM's stakeholders.
[IPO, hasSharesIssuingCompany, GM]
Secondary Issuance Event
Diversifying conglomerate San Miguel Corp. has approved the issuance of secondary common shares and convertible bonds to fund its massive diversification into heavy industries and boost its public float.
[SecondaryIssuance, hasSharesIssuingCompany, San Miguel Corp.]
NEC Corporation (NEC) today announced that at a meeting of the Board of Directors of NEC held on November 6, 2009, NEC resolved as below matters relating to an issuance of new shares and a secondary offering of NEC shares.
[SecondaryIssuance, hasSharesIssuingCompany, NEC Corporation]
Stock Split Event
Quantum Ltd. shareholders approve stock split.
[StockSplit, hasSharesIssuingCompany, Quantum Ltd.]
Timber company Rayonier Inc. said Monday its board of directors approved a three-for-two stock split.
[StockSplit, hasSharesIssuingCompany, Rayonier Inc.]

As you can see in Table 14, there are precisely two types of properties that needs to be stored about any of the above events i.e. a company name and in some cases amount of money involved. Hence the only data member that is added to the basic Event class is money and then the class is sufficient to handle any type of stock event. Stock event respective Java class is shown in Figure 15.

```
StockEvent extends Event
{
    Money money
}
```

Figure 15 – Stock Event Java Class

Work Event Category

Work events come from input text files with the name format '*WorkEvent_xxx.txt*' where *xxx* is the name of any of the specific work events. Specific work events are of three types as shown in Table 15.

Table 15 – Work events input file format

Company Force Majeure Event
A subsidiary of TransCanada Corp. declared a force majeure Thursday.
[CompanyForceMajeure, hasWorkIssuesCompany, TransCanada Corp.]
Reliance Power Transmission's arm Talcher-II Transmission Company has served a force majeure notice to electricity distribution utilities in Andhra Pradesh, Karnataka, Kerala, Tamil Nadu and Orissa.
[CompanyForceMajeure, hasWorkIssuesCompany, Reliance Power]
Company Labor Issues Event
Labour issue at Honda Manesar plant resolved.
[LaborIssues, hasWorkIssuesCompany, Honda]
But the more important flaw here is that the reason why Boeing might have judged its decision to move production to South Carolina "best for its shareholders" was that it didn't think it violated labour law to flee your union.
[LaborIssues, hasWorkIssuesCompany, Boeing]

Table – 15 continues

Company Layoffs Event
Spanish company Telefonica. The planned layoffs, which will affect nearly 20% of the company's 35,000 Spanish employees, are part of Telefonica's strategy to increase profitability at its Spanish division.
[CompanyLayoffs, hasWorkIssuesCompany, Telefonica]
One of Silicon Valley's technology giants, Cisco Systems, could be just weeks away from one of the biggest layoffs in company history.
[CompanyLayoffs, hasWorkIssuesCompany, Cisco Systems]

All the companies in the above table had different sort of work-related issues e.g. Cisco Systems had some layoffs where as Honda had some labor issues. However across events only one kind of information needs to be stored which is the company name, for which the attribute is inherited from the super class *Event*. Like Patent and Bankruptcy events, Work event does not need any data member other than the one it inherits from its super class.

4.4 Ontology Interfacer

Ontology Interfacer is the central component from where all the method calls are initiated to different components of the API. The architecture and data flow of the API components was illustrated in Figure 6 above. Figure 16 below provides a UML sequence diagram of the interaction between the API components.

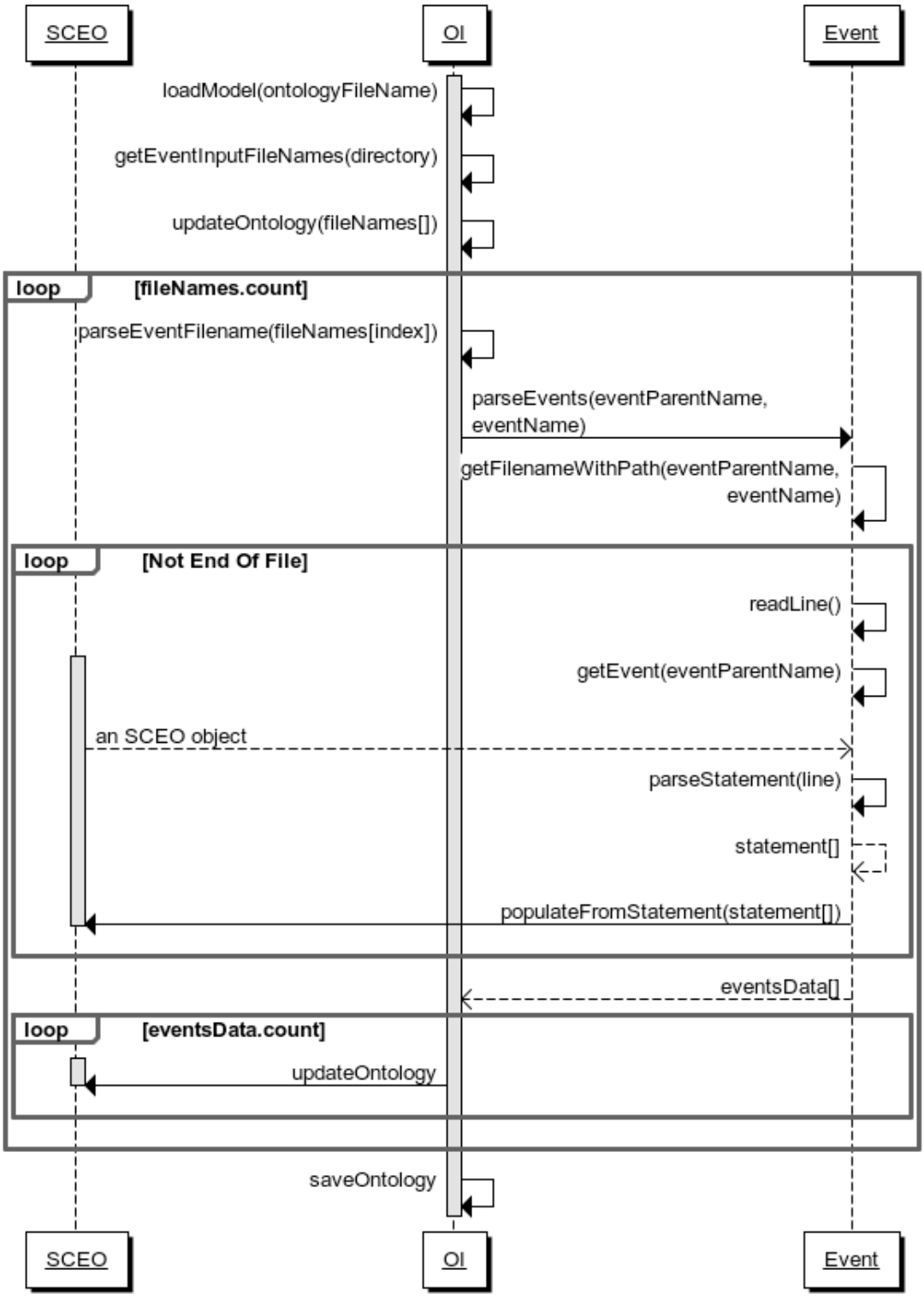

Figure 16 – Sequence diagram of API, OI is instance of Ontology Interfacer and SCEO represents any instance of a sub-classed event object

The first interaction with Jena API occurs through OI with a method called *loadModel* that loads the ontology file into the memory. Loading of ontology into memory is a simple process of passing ontology file as an input stream to the

method of an object provided by Jena API. The loaded ontology is referred to as *Ontology Model*.

All the data input text files of the format that was explained in the previous section are expected to be present under the *'input'* directory. The *input* directory can only contain data text files of one of the formats described above. OI calls the method *getEventInputFileNames* to read names of all files in the '*input'* directory, so that it can start processing each file one by one. Once the file names have been read *updateOntology*, present in OI, is invoked whose pseudo-code is provided in Figure 17*.* It is the main run-loop method of the entire API.

```
1   O ← CoProE ontology
2   Update-Ontology(filenames[])
3     FOR fileNo ← 0 to length[filenames]
4       F ← filenames[fileNo]
5       F-Info ← call Parse-Event-Filename(F)
6       eventsData ← call static method Parse-Events(F-Info) on Event
7       FOR eventNo ← 0 to length[eventsData]
8           E ← eventsData[eventNo]
9           call overridden method Update-Ontology(F-Info) on E
10      ENDFOR
11    ENDFOR
12  Save-Ontology(O)
13  END
```

Figure 17 – Pseudo-code for OI's Update-Ontology method

If we were to map pseudo-code in Figure 17 to the sequence diagram in Figure 16, *updateOntology* starts before the outer loop and ends at the last method call to OI**.** It is the most important method of OI as it is directly or indirectly responsible for calling all methods of other components that in turn do various tasks.

OI starts the iteration (line # 3) over the filenames it had read previously in order to parse out two types of names from it; one we refer to as the *event category name* and the other as *event type name*. For example, the file

'*PatentEvent_PatentFiling.txt*' has the *event category name PatentEvent* and the *event type name* is *PatentFiling*. This filename information is of utmost importance to OI, as without this information API would not be able to tell what kind of data is present inside the current file, thus what kind of SCEO to instantiate. The type of SCEO to create is the same as *event category name*.

The OI, in turn, calls a static method on the *Event* class passing this file information as parameter and in return gets an array of SCEOs called *eventsData* (line # 6). The inner workings of this static method are explained in Section 4.5. OI iterates (line # 7) over *eventsData* and calls overridden method *updateOntology* on each SCEO. The relationship of *updateOntology* for event object is further explained in Section 4.5.

After the execution of line # 9, OI is aware that the ontology model has been successfully updated with data that was present in the SCEO. Finally at line # 12 OI calls a method to write the ontology model back to the ontology file on the hard drive. This action is performed after the file iteration is completed and all the data parsed from the input files updated into the ontology model. This step assures that the changes made to ontology model are persistent.

4.5 Event Class

The Event *class* is akin to middle tier component of the CoProE API. It is an abstract class, which implies that one cannot create an object of the type *Event*. The abstraction restriction was enforced on the *Event* class so that any new event added to the CoProE API is able to expose its interface for other components and yet maintain customization. It is the super class of all event objects, thus the term 'sub-classed event objects'. All of the SCEO must implement the abstract methods of the *Event* class to become functional part of the CoProE API.

The instantiation of a SCEO and its population with data occur through the *Event* class using static and overridden methods respectively. Analyst Event, Company Collaboration Event and Product Event are some of the SCEOs that have been explained in detail in Section 4.6. As explained above, the *Event* class is only

composed of two data members (*id* and *companyName*), two abstract methods (*updateOntology* and *populateFromStatement*) and a number of static and non-static utility methods. The data member *id* serves as a unique identifier of every event object that is instantiated and inserted into the ontology by the API.

Every SCEO must provide the implementation of the following two abstract methods:

A) abstract void populateFromStatement(String[] statement)

B) abstract void updateOntology(OntModel model, String eventName)

The *statement* parameter in method A is a string array containing an RDF statement with the RDF subject, predicate and object stored at array indexes 0, 1 and 2 respectively. The method in A is responsible for populating event object with data provided in the statement given as the parameter. Method B updates ontology model with data encapsulated in the event object. The method calls for methods A & B are shown in Figure 16, in the first and the second inner loop respectively. Several overridden implementation of these methods are discussed in Section 4.6.

The remaining methods and the most important non-static method signatures are given below.

C) void updateOntologyWithCompany(OntModel model, OntClass thingClass, Resource companyResource, String cName)

D) void updateOntologyWithEventAndProperty(OntModel model, Resource propertyValue, String eventName, String oProperty)

E) void updateOntologyWithEventAndMoney(OntModel model, String eventName, Money money)

The *eventName* parameter in methods B, D and E is the name of a specific business event e.g. *ProductRelease*, *PatentFiling*, *Merger* etc and the *model* parameter in methods B, C, D and E is an object that represents the ontology. The names of the methods are fairly self-explanatory regarding the task of each method. All these methods update ontology model using Jena API interfaces and

their methods. This is one of the two points where API interacts with Jena API; the first point is in OI. Figure 18 shows the body of method C, D & E.

Method C

```
//if company doesn't exists in ontology
if (!model.containsResource(companyResource))
{
    OntClass companyClass = model.getOntClass(Ontology.ONTOLOGY_NS + "Company");
    companyClass.createIndividual(Ontology.NEWSEVENT_NS + cName);
    thingClass.createIndividual(Ontology.NEWSEVENT_NS + cName);
}
```

Method D

```
OntClass eventClass = model.getOntClass(Ontology.ONTOLOGY_NS + eventName);
Individual eventInd = eventClass.createIndividual(
        Ontology.NEWSEVENT_NS + eventName + id);
ObjectProperty objectProperty = model.createObjectProperty(
        Ontology.NEWSEVENT_NS + oProperty);
eventInd.addProperty(objectProperty, propertyValue);
```

Method E

```
if (money.currency != null)
{
    DatatypeProperty dataProperty = model.createDatatypeProperty(
            Ontology.NEWSEVENT_NS + "hasCurrency");
    eventInd.addProperty(dataProperty, money.currency);
}

if (money.unit != null)
{
    DatatypeProperty dataProperty = model.createDatatypeProperty(
            Ontology.NEWSEVENT_NS + "hasUnit");
    eventInd.addProperty(dataProperty, money.unit);
}

if (money.value != null)
{
    DatatypeProperty dataProperty = model.createDatatypeProperty(
            Ontology.NEWSEVENT_NS + "hasValue");
    eventInd.addProperty(dataProperty, money.value);
}
```

Figure 18 – Jena methods for updating ontology, *OntClass*, *Individual*, *ObjectProperty*, *DatatypeProperty* are interfaces provided by Jena API and *createIndividual*, *addProperty* are methods belonging to those interfaces

The non-static methods shown in above figure are logically available to all inheriting event classes and are extensively used by all of them to update ontology model. Method C adds the company passed as the parameter into the ontology in case it does not exist. The parameter *cName* is used as the name of the new company. Method D updates the ontology with an event and the related object property, *oProperty*, which can have values such as *hasPatentHoldingCompany*, *hasWorkIssuesCompany* etc. Method E updates the

ontology with event and its related money value; the *money* parameter is a custom object that contains all information regarding sums of money (value, currency etc.) that has to be added into the ontology.

Two static methods signatures from the *Event* class are provided below:

F) static Event getEvent(String eventType, long id)

G) static String[] parseStatement(String line)

As it was mentioned in the previous section, OI parses two types of names from a file: the *event category name* and the *event type name*. Method F simply returns an instantiated SCEO based on what was specified to it in the *eventType* parameter. *eventType* and *event category name* are identical. Method G is a utility method to parse a line read from text file. For example:

[LaborIssues, hasWorkIssuesCompany, Honda]

is returned by the method as an array of String[0,1,2] = {“LaborIssues” , “hasWorkIssuesCompany” , “Honda”}. The most important static method that the *Event* class provides and which acts as the only interface between OI and the *Event* class is *parseEvents* (see Figure 17 line # 6). Pseudo-code of the *parseEvents* method is shown in Figure 19.

```
1   Parse-Events (eventParentName, eventName)
2     F ← call Get-Filename-With-Path(eventParentName, eventName)
3     E ← call static method Get-Event(eventParentName)
4     Insert(E) to eventsData
5     WHILE NotEndOfFile (F)
6         line ← ReadLine(F)
7         IF line IsEmpty()
8           E ← call static method Get-Event(eventParentName)
9           Insert(E) to eventsData
10          continue
11        ENDIF
12        S[subject, predicate, object] ← Parse-Statement(line)
13        call overridden method Populate-From-Statement(S) on E
14    ENDWHILE
15  RETURN eventsData
```

Figure 19 – Pseudo-code for *Event* class ParseEvents static method

There are four key responsibilities of *parseEvents:*

1) To determine the type of the SCEO to create (line # 3, method F), *parseEvents* creates the same number of event objects as empty lines found; an empty line is an indicator for start of a new event data. *ParseEvents* inserts the line into the *eventsData* list. If the *event category name* is *ProductEvent* then *ProductEvent.class* object will be instantiated. If the *event category name* is *AnalystEvent* then *AnalystEvent.class* will be instantiated and similarly for the other event categories.

2) Read the whole text file one line at a time (line # 6)

3) Parse the data read from the file (line #12, method G). The method guarantees that each line in the text file is an RDF triplet. Hence, it safely calls the parse method on the line read from the text file to return the subject, predicate and object as tokens.

4) And finally, populate the event object it instantiated with data read from the input text file (line #13, method A). Line # 15 returns the *eventsData* to OI.

4.6 Sub-classed Event Objects

All the SCEOs combined form the third component of the API. OI directly invokes methods on SCEOs that are provided to it as a list by the *Event* class. As was discussed in previous section, there are two abstract methods whose implementation every event object must provide. In simple terms *populateFromStatement* is related to reading existing data and *updateOntology* is related to writing new data.

As every event type has different type of data to read and write, only an event itself can be aware of how to process its data. The main task of *populateFromStatement* is to populate the data members in the Java classes discussed in Section 5.3. The main task of *updateOntology* is to take data from these data members and insert it into the CoProE ontology. The *populateFromStatement* method is invoked from the *parseEvents* method in the *Event* class (see first inner loop in Figure 16) where as *updateOntology* is invoked from OI (see second inner loop in Figure 16). The following section will discuss implementation details of these two methods for three distinctive events categories.

Analyst Event Object

Analyst event is responsible for reading and writing three specific analyst events types: *AnalystEarningsEstimate*, *CompanyEarningsAnnouncement* and *CompanyEarningsGuidance*. The pseudo-code of *populateFromStatement* for Analyst event is provided in Figure 20.

```
1  S ← statement provided by Parse-Events method
2  Populate-From-Statement(S)
3      IF S[predicate] EQUALS "hasAnalyst" OR
4      S[predicate] EQUALS "hasAnnoucingCompany"
5          E[companyName] ← S[object]
6      ELSEIF S[predicate] EQUALS "hasMoney"
7          E[money] ← static method Parse-Money(S[object]) on Event
8      ELSEIF S[predicate] EQUALS "hasAnalysedCompany"
9          E[analysedCompany] ← S[object]
10     ELSEIF S[predicate] EQUALS "hasParticipant"
11         E[participant] ← S[object]
12     ELSEIF S[predicate] EQUALS "usesFinancialMetric"
13         E[financialMetric] ← static method Parse-Financial-
14         Metric(S[object]) on Event
15     ELSEIF S[predicate] EQUALS "hasFinancialTrend"
16         E[financialTrend] ← S[object]
17     ELSEIF S[predicate] EQUALS "hasPercent"
18         E[percent] ← static method Parse-Percentage(S[object]) on E
19     ENDIF
20 END
```

Figure 20 – Pseudo-code of Analyst event statement parser method

The logic of the method in Figure 20 is fairly simple and it, in fact, remains fairly similar across the various implementations of this method. An Analyst event itself is aware of the RDF predicates it will encounter in the RDF statement. Hence, it checks for all the possible predicates in the statement and parses out the required information from the statement accordingly.

Let us consider line # 8 as an example. On finding a predicate '*hasAnalysedCompany*', the method directly reads the analyzed company name from the statement and stores it in a data member. Compare that to line # 17, where a '*hasPercent*' attribute is found and the method calls a specific utility method to parse percentage value from the statement. A similar case is present at line # 13 where the aim is to parse some financial metric information. All the

read data is transferred to the ontology by *Analyst* event *updateOntology* implementation shown in Figure 21.

```
1  O ← CoProE ontology
2  Update-Ontology(O, eventName)
3    C ← Update-Ontology-With-Company(E[companyName])
4    R ← Add a new eventName Resource in O
5    IF eventName EQUALS ANALYST_EARNINGS_ESTIMATE
6      IF companyName NOT NULL
7        Update-Ontology-With-Event-Property(eventName, C,
8        "hasAnalyst")
9      ENDIF
10     IF analysedCompany NOT NULL
11       AC ← Update-Ontology-With-Company(E[analysedCompany])
12       Update-Ontology-With-Company-Type(E[analysedCompany],
13       "AnalysedCompany")
14       Update-Ontology-With-Event-Property(eventName, AC,
15       "hasAnalysedCompany")
16     ENDIF
17     IF participant NOT NULL
18       P ← Update-Ontology-With-Company(E[participant])
19       Update-Ontology-With-Event-Property(eventName, P,
20       "hasParticipant")
21     ENDIF
22   ELSEIF eventName EQUALS
23   COMPANY_EARNINGS_ANNOUNCEMENT
24      Update-Ontology-With-Company-Type(E[companyName],
25      "AnnouncingCompany")
26      Update-Ontology-With-Event-Property(eventName, C,
27      "hasAnnouncingCompany")
28      IF percent NOT NULL
29        Add-Property("hasPercent") to R
30      ENDIF
```

Figure – 21 continues

```
31    ELSEIF eventName EQUALS COMPANY_EARNINGS_GUIDANCE
32        Update-Ontology-With-Company-Type(E[companyName],
33        "AnnouncingCompany")
34        Update-Ontology-With-Event-Property(eventName, C,
35        "hasAnnouncingCompany")
36    ENDIF
37    IF money NOT NULL
38        Update-Ontology-With-Event-Money(eventName, money)
39    ENDIF
40    IF financialMetric NOT NULL
41        FM ← Add a new financialMetric Resource in O
42        Update-Ontology-With-Event-Property(eventName, FM,
43        "usesFinancialMetric")
44    ENDIF
45    IF financialTrend NOT NULL
46        FT ← Add a new financialTrend Resource in O
47        Update-Ontology-With-Event-Property(eventName, FT,
48         "hasFinancialTrend")
49    ENDIF
50    IF quarter NOT NULL
51      Add-Data-Property("hasQuarter", quarter) to R
52    ENDIF
53    IF year NOT NULL
54      Add-Data-Property("hasYear", year) to R
55    ENDIF
56  END
```

Figure 21 – Pseudo-code of the Analyst event updateOntology method

As there are three types of specific analyst events that are to be expected, the method in Figure 21 first checks the type of event that has been passed to it. Based on this information it knows that certain data members would have values that needs to be inserted into the ontology. The method uses its inherited methods for updating the ontology. For example, at line # 34 it invokes a method to add announcing company property to the event instance. Towards the end,

the method checks if the value is available for any of the data members that are common across different analyst event and if any of them is found it is inserted into the ontology.

Company Collaboration Event Object

The *Company Collaboration* event class is responsible for reading and writing data related to four specific company collaboration event types: Alliance, BusinessRelation, JointVenture and Merger. The pseudo code for the *populateFromStatement* method is shown in Figure 22.

```
1  S ← statement provided by Parse-Events method
2  Populate-From-Statement(S)
3      IF S[predicate] EQUALS "hasAllyingCompany" OR
4      S[predicate] EQUALS "hasRelatedCompany" OR
5      S[predicate] EQUALS "hasVenturingCompany" OR
6      S[predicate] EQUALS "hasMergingCompany"
7          Insert(S[object]) to E[collaboratingCompanies]
8      ENDIF
9  END
```

Figure 22 – Pseudo-code of company collaboration event statement parser method

As one can observe from Figure 22, the implementation of Company Collaboration event is much simpler and shorter compared to the other events. The reason is that the input data for a Company Collaboration event is of far simple nature compared to other events. This is where the flexibility and strength of the API is at its most evident. It allows adapting to different formats of data as long as the relevant SCEO provides the correct implementation. Whatever data is parsed by the method above, it is stored in an array that is later processed by *updateOntology* method in Figure 23.

```
1  O ← CoProE ontology
2  Update-Ontology(O, eventName)
3    FOR companyIndex ← 0 to length[collaboratingCcompanies]
4    C ← Update-Ontology-With-Company(E[companyName])
5    IF eventName EQUALS ALLIANCE
6       Update-Ontology-With-Company-Type(E[collabCompany],
7       "AllyingCompany")
8       Update-Ontology-With-Event-Property(eventName, C
9       "hasAllyingCompany")
10   ELSEIF eventName EQUALS BUSINESS_RELATION
11      Update-Ontology-With-Company-Type(E[collabCompany],
12      "RelatedCompany")
13      Update-Ontology-With-Event-Property(eventName, C,
14      "hasRelatedCompany")
15   ELSEIF eventName EQUALS JOINT_VENTURE
16      Update-Ontology-With-Company-Type(E[collabCompany],
17      "VenturingCompany")
18      Update-Ontology-With-Event-Property(eventName, C,
19      "hasVenturingCompany")
20   ELSEIF eventName EQUALS MERGER
21      Update-Ontology-With-Company-Type(E[collabCompany],
22       "MergingCompany")
23      Update-Ontology-With-Event-Property(eventName, C,
24      "hasMergingCompany")
25   ENDFOR
26 END
```

Figure 23 – Pseudo-code of the Company Collaboration event *updateOntology* method

The logic to update ontology in Figure 23 is fairly simple. The method iterates over the data of all the collaborating companies data. After determining what type of specific event has been passed as parameters, the method it updates the ontology with respective predicate property and company name.

Product Event Object

Product event is responsible for the reading and writing of three specific Product events types: ProductIssues, ProductRecall, ProductRelease. The *populateFromStatement* method for Product event is specified in Figure 24.

```
1  S ← statement provided by Parse-Events method
2  Populate-From-Statement(S)
3      IF S[predicate] EQUALS "hasProvider" OR
4          E[companyName] ← S[object]
5      ELSEIF S[predicate] EQUALS "hasProduct"
6          Insert(S[object]) to E[products]
7      ENDIF
8  END
```

Figure 24 – Pseudo-code of the Product event statement parser method

As can be seen in Figure 24, Product event implementation is also very simple as the only two RDF predicates that can be encountered are '*hasProvider*' or '*hasProduct*'. Due to different format of the data it requires a different kind of implementation for *updateOntology* as shown in Figure 25.

```
1  O ← CoProE ontology
2  Update-Ontology(O, eventName)
3    Update-Ontology-With-Company(E[companyName])
4    Update-Ontology-With-Company-Type(E[companyName],
5     "Provider")
6    R ← Add a new eventName Resource in O
7    FOR productNo ← 0 to length[products]
8      P ← products[productNo]
9      Add(P) to Product Resources in O
10     Add-Property("isProducedBy") to related Company Resource in O
11     Add-Property("hasProduct", P) to R
12   ENDFOR
13 END
```

Figure 25 – Pseudo-code of product event update ontology method

As can be seen in Figure 25, this implementation is quite different from the other two implementations explained earlier in the section. In fact, there is no condition for checking the type of Product event. As they all share the same common predicate (i.e. '*isProducedBy'* and '*hasProduct'*), it does not matter what type of specific product event name has been passed to the method. The only thing that matters is that a new resource for the event name is added to ontology (line # 6). The rest of the actions are performed for that specific event resource for all the products found in the products array.

All the event types whose data input files were described in Section 4.3 have their own implementation for the abstract methods described in this section. The above three parent events have been chosen to demonstrate implementations that are representative for almost all the event types. At the same time, these examples also illustrate the scalable nature of the API.

4.7 Ontology Persistence and Querying

There are two major decisions that had to be taken while designing and developing the CoProE API: one has to do with how to implement data persistence and the other is linked to querying ontology for any given data. Jena provides multiple ways of loading an ontology model from the disk into the memory. For example, ontology can be loaded in memory without or without support for an inference engine. The ontology model can be persisted in a database or a file. Generally, if the changes to ontology are to be persisted, the ontology model is loaded into Jena from a database. Plug-in is available for Jena that enables it to load ontologies from a database [28]. The problem we faced by loading CoProE through database is that the persistent ontology is stored in a database and, thus, cannot be read directly through ontology editors, such as Protégé. In addition, we could not determine any benefit from loading it through database; the more simpler and efficient way was to write the ontology back to the input stream it was loaded from.

Jena provides query support on top of ontology through *SPARQL Protocol and RDF Query Language* (SPARQL) plug-in [29]. SPARQL is suitable if requirement is to construct really extensive and advanced queries. In our case the only query

that is required is checking existence of a particular resource (subject or object in a RDF triplet) and that is handled well with the existing methods in Jena API.

4.8 Testing and Evaluation

The correct working of the API is central to the success of the DAVID system. The API was built according to the requirements of the DAVID system and the end goals of the Towards e-leadership project. The API takes the responsibility of assuring that it conforms to these requirements and goals. Therefore, we followed a certain process to verify the working of the API. The approach we took to test the API is a similar to a *black-box testing* approach. Black-box testing is a method of software testing that tests the functionality of a software as opposed to its internal workings. In the context of our API what it means is that given an input data we expect the CoProE ontology to be correctly updated with certain semantic data. That verifies the functioning as well as the internal methods of the API.

In order to have an error-free integration and communication with the system, the API works under certain strict assumptions and if these assumptions are violated the API will fail to function:

1. Event filenames must follows format: *ParentEvent_EventName.txt* e.g. *AnalystEvent_CompanyEarningsAnnouncement.txt*. It cannot be provided in any other format, as the API has to parse event categories and event types.
2. The name of file should end with RDF *resource* present in Ontology and start with event supported in the API e.g. *PatentEvent_PatentIssuance.txt* is correct but *PatentEvent_Issuance.txt* is wrong, *CompanyGrowth_CompanyExpansion.txt* is correct but *CompanyGrowth_Expansion.txt* is wrong.
3. Input data files should contain event *resource* present in Ontology e.g. *PatentIssuance* is correct *PatentIssue* is wrong.
4. Only events input files should be in *Ontology.INPUT_FILES_DIRECTORY*
5. An empty line must separate events in a text file.
6. Path to an RDF *resource* can be full path or not i.e. (*Event/ProductEvent/ProductRelease*, *hasProduct*, *Computer_Services#HP_Cloud_System*) or (*ProductRelease*, *hasProduct*, *HP_Cloud_System*).

7. Data is case-sensitive e.g. if a certain product *hasType* is *Computer_services* then text file cannot specify *Computer_Services*, even a mismatch of one character will create problem.

Next, to conclude our discussion on CoProE API development, we will discuss how the API was tested and evaluated. Figure 26 provides a compact view of the events hierarchy and the event types.

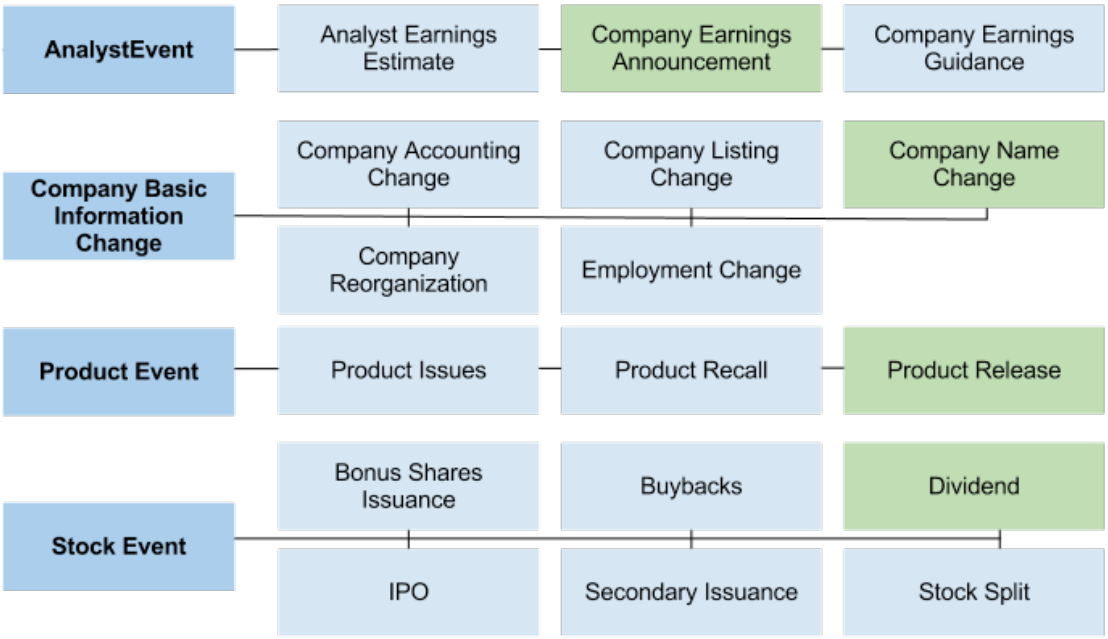

Figure 26 – CoProE ontology events hierarchy with tested event types

The green boxes in the figure represents events whose testing details are elaborated in this section. CoProE ontology can be accessed in two ways: Protégé [22] and through the API that we developed. The insertion of all business events occurred through the API and the updated ontology was viewed with Protégé to validate the correctness of the ontology update. It is much more efficient and reliable to verify correctness of ontology update manually compared to writing some sort of logical unit tests. A logical unit test would be suitable if we wanted to test a specific method.

Figure 27 shows an instance of Company Earnings Announcement event in text file form and after ontology update.

[CompanyEarningsAnnouncement, hasAnnoucingCompany, Honda]
[CompanyEarningsAnnouncement, usesFinancialMetric, quarterly operating profit]
[CompanyEarningsAnnouncement, hasFinancialTrend, fall]
[CompanyEarningsAnnouncement, hasPercent, 90 percent]

Figure 27 – Validation of a Company Earnings Announcement event

As shown in Figure 27 there are four RDF statements in the input text and their resultant statements are present in the screenshot from Protégé. This means that the API has converted '*quarterly operating profit'* to '*metricQuarterProfit'* as that is an existing *resource* in ontology and has also parsed the percentage value.

Let us consider another example of Company Earnings Announcement event in Figure 28.

[CompanyEarningsAnnouncement, hasAnnoucingCompany, Rosneft]
[CompanyEarningsAnnouncement, usesFinancialMetric, 2Q 2011 net income]
[CompanyEarningsAnnouncement, hasMoney, $2.831 billion]

Figure 28 – Validation of a Company Earnings Announcement event

While the data in Figures 27 and 28 represent the same event type, the predicate values are considerably different. It has a new predicate '*hasMoney*' that the API has successfully tokenized and inserted the appropriate data properties. The API has converted '*2Q 2011 net income*' to '*metricQuarterIncome*' and added data properties for *hasQuarter* & *hasYear*. The API is capable of parsing financial and monetary metrics. If we look at example of Dividend event in Figure 29 it also has a '*hasMoney*' property.

[Dividend, hasMoney, $0.23]
[Dividend, hasDividendPayingCompany, Ameriprise Financial]

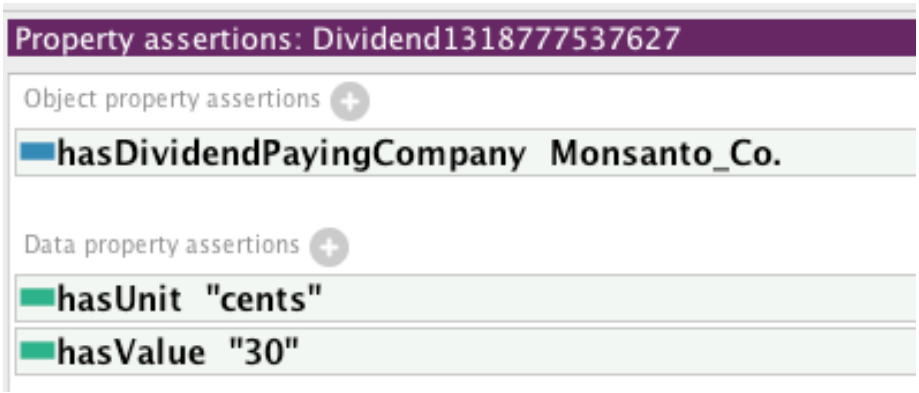

Figure 29 – Validation of Dividend event

In case of Dividend event in Figure 29 the API is able to interpret that the amount of money is really small and represents few cents. It updates the ontology accordingly with '*30* and '*cents*'.

Figure 30 shows an instance of Company Name change event in text file form and after ontology update.

```
[CompanyNameChange, hasOldCompanyName, Swaraj Mazda Limited]
[CompanyNameChange, hasNewCompanyName, SML Isuzu Limited]
```

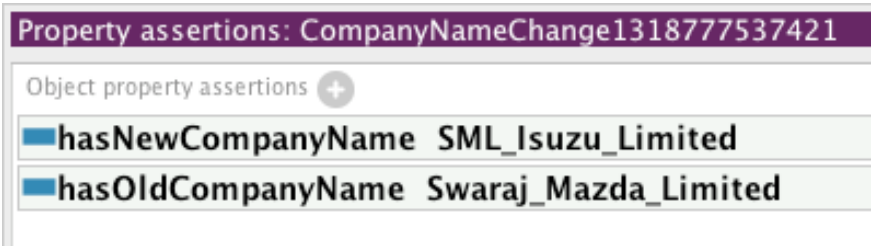

Figure 30 – Validation of a Company Name Change event

The resulting properties in Protégé are pretty much a direct map of the statements in input text file. As the new company name is also an ontology *resource* thus it falls under object property assertions. Figure 31 shows another simple case of direct mapping of properties from input file.

```
[ProductRelease, hasProduct, Wildfire S]
[ProductRelease, hasProvider, HTC Corporation]
```

Figure 31 – Validation of a Product Release event

As the API is heavily built on code reuse principles, for all events same methods are used for updating ontology but with different method parameter values. This greatly reduces the chances of error in API and leads to efficient testing.

The testing and validation process discussed above was performed for all event categories and types, however, for avoiding being redundant we limited the discussion above to only a representative sample of all events. In order to have a better understanding of the scale of testing the statistics of test data are plotted in Figure 32.

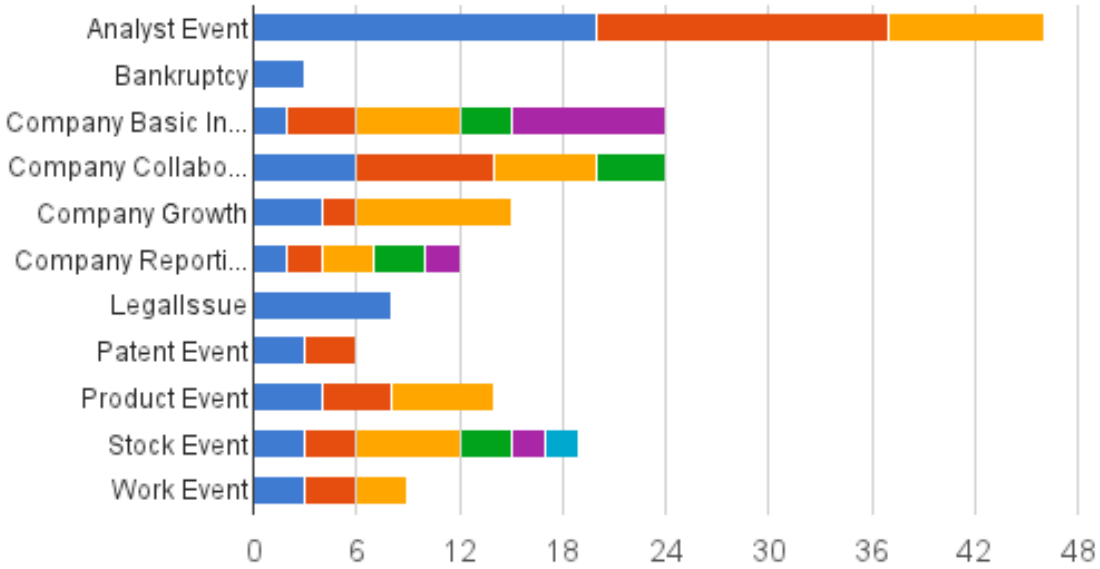

Figure 32 – Test data statistics for event categories and the event types belonging to each of them

As has been mentioned before, events are divided into 11 event categories that, with the exception of Bankruptcy, are divided into several event types. In Figure 32, each subcategory is represented by a unique color within a single bar and the *x*-axis represents the number of RDF statements validated for each event type. For example e.g. Patent Event has two event types (represented by blue and red color) and three test statements for each event type. There were a total of 180 test RDF statements. This data was provided to us in the form of input text files that the API is designed to parse. The input data was produced automatically by the BEECON entity and event extraction system from business articles.

Testing was a continuous process performed in parallel during the development of the API. On adding each new event to the API, tests were performed and ontology content and resources evaluated by using Protégé. Any errors or inconsistencies that were noticed during the testing process were readily visible

in Protégé. The cause for the error would be located and fixed before continuing with further development. This testing methodology allowed us to build a high quality ontology API without much previous knowledge and experience with building SW applications and ontologies.

5 Conclusion

In this thesis, we have extended CoProE ontology that was developed as part of a larger TM system for collecting and analyzing business texts. We presented the concepts behind making ontology and the internal structure of CoProE. The TM system needs a flexible data store environment (i.e. domain knowledge base) from where it can efficiently retrieve and analyse business data with the ultimate goal of helping business leaders to reach better decisions. We achieved that by inserting data about five Finnish companies and their products into CoProE. Furthermore, in order to have the ability to retrieve business events and store into CoProE we demonstrated the development and evaluation of a Java API.

The first research objective was to "*Collect data about companies, products and related information to populate the CoProE ontology*". This led us to develop a comprehensive methodology for collecting information about companies and their business operations primarily using a questionnaire-based survey. Knowledge about each company profile, products, services, patents, legal issues, competitors provide great value for BI creation. This information allows the system to leverage the stored information in analyzing input documents and for carrying new research or innovating new systems for BI.

The second research objective was to "*Modify CoProE in case it is not capable to store gathered business data*". The objective was to explore the design of CoProE and understand the capability of the ontology for storing business knowledge. The structure of CoProE revealed by our analyses and data collection showed that CoProE is fully capable of supporting our surveyed business data. Hence, no changes were needed in the structure of the ontology.

The third research objective was to "*Design and implement a Java API for searching and manipulating CoProE*". We did that by successfully building and

integrating a highly scalable and flexible API. The integration of CoProE as a domain knowledge base is crucial for the DAVID system to find and analyze BI data. The API supports storing entities and events and supports the dynamic nature of business data gathered by the TM system. As the amount of business knowledge in the system gradually grows, intelligent filtering of CoProE, knowledge sharing, and performance metrics will lead the way for more powerful BI.

The last research objective was to "*Verify the working of the API and test it under a suitable methodology*". We evaluated the API using established testing process and realistic input data. The evaluation results clearly showed that it conforms to the requirements that were set in the beginning of this thesis project. As more business events are added to the ontology, the testing approach outlined would serve to maintain the stability and quality of the API.

These conclusions highlight the impact and significance of extending CoProE. Applying the processes and procedures employed in this thesis to other BI system has potential for further understanding of the commonalities and peculiarities of developing intelligent data systems for future BI systems. Two implications for future research can be derived from this work:

1. Knowledge stored in CoProE can be refined to include further information about companies. The ideal case would have been if all companies fully completed the questionnaire and specified all information. However, from our point of view, the participation of companies was relatively low. This created a certain inadequacy for us, but we were able to compensate for that by finding sufficient information about participating companies on the Internet and utilizing that in CoProE as additional background knowledge. From the perspective of comprehensive competitor analysis, it would be really valuable to have more information about the competitors and their business operations inserted into CoProE. Detailed information on patents and competitor patents would also be greatly beneficial for investigating research and future activities

among companies. Moreover, adding data about more companies in different business domains could greatly increase the value of CoProE.

2. The API was designed from the perspective of future extension and it is highly scalable to support any further classification of business events. The API is based on accepting input from plain text files, however, due to its employed programming principles of *inheritance* and *polymorphism* it can adapt to other format of inputs e.g. real time data feed. That makes it viable as a plug and play component in other BI systems or if the existing communication interfaces are modified. The API could be improved to minimize the assumptions it makes about the format of incoming data. The testing of the API was stable however further improvements could be made in this area by testing it with larger datasets and more real-world business events. As there would be larger amount of data, automated testing procedures could be investigated and employed.

Appendices

Appendix A

E-leadership project **Questionnaire of company, product and competitor information**		
Note: This form is for you if your company is participating in the Towards e-leadership project. We appreciate the time taken out to fill this form. Filling the questionnaire is voluntary, however, of fundamental importance for the project. We need this information as background knowledge for the automatic business intelligence analyses. The more information you provide us, the better results your company can expect from the system developed in the project. ***All information shared is under the non-disclosure agreement included in the project agreements, and will be kept highly confidential.*** *You can leave any sections of the questionnaire blank, if you feel that it cannot be disclosed.* *Thanking you in advance for your time and effort!*		
All input fields support **multiline** text; feel free to format response on multiple lines. For your assistance we have also provided a sample form that has been filled in partially. Please have a look at it to understand what type of input is expected from you in different sections.		
SECTION 1 **COMPANY PROFILE**		
A Basic information		
Name		
Industry		
Founded (year)		
Headquarters location		
Other Key locations		
Main Market Area(s)	☐ Worldwide	☐ Specific

Parent Company (if your company is a subsidiary)		
How would you describe your company organizational structure? ☐ Functional ☐ Divisional ☐ Functional + Divisional Others		
B Personnel		
Number of Employees		
Key People	Name	Position
C Financial data		
Stock symbol (if company shares are traded in a stock exchange)	Public:	
Revenue (last fiscal year)		
Operating Income		
Profit		
Total Assets		
Total Equity		
Subsidiaries		
If required would you be willing to provide more information about your subsidiaries ☐Yes ☐No		
Describe in few words what does your company aim to achieve		
SECTION 2		

PRODUCTS AND SERVICES	
A Products	
Do you have an online product catalog?	☐ No, read next question ☐ Yes Please provide a link or attach a copy of your product catalog in your email response to this questionnaire. Link to the catalog:
Do you have an offline product catalog? ☐ No, see next question ☐ Yes, Please provide contact details of who to contact in order to receive a copy:	
Name	
Email	
Phone	
Address	
Products Listing (if you do not have a catalog): Name	Quantity Sold
B Services	
Do you have an online services catalog?	☐ No, read next question ☐ Yes Please provide a link or attach a copy of your services catalog in your email response to this questionnaire. Link to the catalog:
Do you have an offline services catalog? ☐ No, see next question ☐ Yes, Please provide contact details of who to contact (if different from	

above):	
Name	
Email	
Phone	
Address	
Services Listing (if you do not have a catalog): Names	Number of Users
C Key products and services	
What are your key products that keep your company sustainable?	
What are your key services that keep your company sustainable?	
What are the advantages that your products/services offer to your customers that your competitors are unable to provide?	
SECTION 3 **LEGAL INFORMATION**	
List all your key patents/copyrights/trademarks etc:	
Reference	Any company that has importance init
If applicable, please specify any online link or attach a document describing details of your company patents/copyrights/trademarks etc. Online link: ☐Or I will attach/send a documents describing details of company patents/copyrights/trademarks or contact the following personnel (if different from above):	
Name	
Email	
Phone	

Address	

Do you have any legal disputes that you like us to know about:

☐ No

☐ Yes:

Please, describe:

SECTION 4

COMPETITOR INFORMATION

A Competitors and their products and services

List the name of all competing products and their respective manufacturer:

Product Name	Manufacturer

List the name of all competing services and their respective provider:

Service Name	Provider

B Competitors' patents

List the key patents/copyrights/trademarks of your competitors that are of importance to you:

Name	Company

C Legal disputes with competitors

Do your competitors have any important legal disputes that you like us to know about

☐ No

☐ Yes:

Company	Dispute Description

SECTION 5

CLIENTS/CUSTOMERS INFORMATION

A Customers

Describe in few words the typical customer of your product/services:		
List the names of your key clients:		
B Product and services reviews		
Online resources of customer reviews of your products/services:		
C Customer feedback		
Does you company regularly collect feedback from customers: ☐ No. Please, skip to Section 7 ☐ Yes:		
Please, describe what kind of customer feedback your company collects: In which format are the customer feedbacks and how are they stored (for example, on paper, in an information system): The number of archived customer feedbacks:		
SECTION 6 **MISCELLANEOUS INFORMATION**		
A Subcontractors and partners		
List any key companies/contractors/manufacturers/suppliers that <u>you are dependent on</u>:		
Company Name	Company Relationship	Location
List any key companies/contractors/manufacturers that are <u>dependent on you</u>:		
Company Name	Company Relationship	Location
Are there any other companies who you network with that you like to tell us about?		

What areas does the company aspire to continue to invest, develop and improve in?
B Other sources of information
Are there any other sources of business intelligence (information about your company, products, competitors, customers etc) that you like to tell us about?
We thank you once again for taking out the time to fill in this form!

References

1 Wray, Richard. "Internet Data Heads for 500bn Gigabytes." *Guardian | Guardian.co.uk*. 18 May 2009. Web. 27 Oct. 2010. <http://www.guardian.co.uk/business/2009/may/18/digital-content-expansion>.

2 *Siegel, David. "The History of Information", Vimeo, Video Sharing For You*. Web. 27 Oct. 2010. <http://vimeo.com/11117216>.

3 Berners-Lee, Tim, James Hendler, and Ora Lassila. "Publishing on the Semantic Web." *Scientific American* (2001): 29-37. Print.

4 "Semantics." *Wikipedia, the Free Encyclopedia*. Wikipedia. Web. 27 June 2011. <http://en.wikipedia.org/wiki/Semantics>.

5 Gaudin, By Sharon. "In Schmidt's Vision, Google Will Search Before You Even Ask." *Reviews and News on Tech Products, Software and Downloads - PCWorld*. 30 Sept. 2010. Web. 27 Oct. 2010. <http://www.pcworld.com/businesscenter/article/206665/in_schmidts_vision_google_will_search_before_you_even_ask.html>.

6 Hearst, Marti. *What Is Text Mining?* SIMS, UC Berkeley, 17 Oct. 2003. Web. 09 June 2011. <http://people.ischool.berkeley.edu/~hearst/text-mining.html>.

7 Kakkonen, T., Mufti, T.: Developing and Applying a Company, Product and Business Event Ontology for Text Mining. *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, Graz, Austria, 2011.

8 Cearley,, David W., Whit Andrews,, and Nicholas Gall. *Finding and Exploiting Value in Semantic Technologies on the Web*. Publication no. G00148725. Gartner, 9 May 2007. Web. 1 Nov. 2010. <http://www.gartner.com/DisplayDocument?ref=g_search&id=505304>.

9 Hepp, Martin: GoodRelations: An Ontology for Describing Products and Services Offers on the Web, Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy, September 29 - October 3, 2008, Springer LNCS, Vol 5268, pp. 332-347.

10 Iskold, Alex. "ReadWriteWeb." *ReadWriteWeb*. 6 May 2010. Web. 07 Mar. 2012. <http://www.readwriteweb.com/archives/does_facebook_really_want_a_semantic_web.php>.

11 "Annotations Overview." *Annotations Overview | Dev.twitter.com*. Twitter. Web. 1 Nov. 2010. <http://dev.twitter.com/pages/annotations_overview>.

12 Ganz, Steve. "LinkedIn Launches HResume." Web log post. *Steve Ganz Blog*. 26 Jan. 2007. Web. 1 Nov. 2010. <http://steve.ganz.name/blog/2007/01/linkedin-launches-hresume.html>.

13 Ha, Inay, Kyeong-Jin Oh, Myung-Duk Hong, Yeon-Ho Lee, Ahmad Nurzid Rosli, and Geun-Sik Jo. "Ontology-driven Visualization System for Semantic Searching." Multimedia Tools and Applications (2011). 3 Nov. 2011. Web. 21 Mar. 2012. <http://www.springerlink.com/content/q72891n383596378/>.

14 Lösch, U., Nikitina, N.: The newsEvents Ontology – An Ontology for Describing Business Events. In: 8th International Semantic Web Conference, 1st Workshop on Ontology Design Patterns, Washington DC, USA (2009)

15 "United Nations Standard Products and Services Code." *UNSPSC Homepage*. UNDP. Web. 27 June 2011. <http://www.unspsc.org/>.

16 Kakkonen, T., Mufti, T.: Developing and Applying a Company, Product and Business Event Ontology for Text Mining. *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, Graz, Austria, 2011.

17 Bhowmick PK, Roy D, Sarkar S, Basu A (2010) A framework for manual ontology engineering for management of learning material repository. J Comput Sci Appl 7(2):30–51

18 Crowder RM, Wilson ML, Fowler D, Shadbolt N, Wills G, Wong S (2009) Navigation over a large ontology for industrial web applications. IDETC'09: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, San Diego, USA, pp 1333–1340

19 YongYue C, HuoSong X (2009) Research on knowledge extraction and visualization in knowledge retrieve. International Conference on Intelligent Human-Machine Systems and Cybernetics, vol 2, Hangzhou, China, pp 66–69

20 "ISTweb - Content - Knowledge & Content Technologies - Projects - SALERO." *EUROPA - CORDIS: Community Research and Development Information Service*. 28 Oct. 2010. Web. 27 June 2011. <http://cordis.europa.eu/ist/kct/musing_synopsis.htm>

21 R. C. LaBrie and R. D. S. Louis, "Dynamic hierarchies for business intelligence information retrieval," International Journal of Internet and Enterprise Management, vol. 3, no. 1, pp. 3–23, 2005.

22 *Protege*. Computer software. *Protege Wiki*. Stanford Center for Biomedical Informatics Research. Web. 27 June 2011. <http://protegewiki.stanford.edu/wiki/Main_Page>.

23 "Learn About Java Technology." *Learn about Java Technology*. Web. 26 Feb. 2012. <http://www.java.com/en/about/>.

24 "Eclipse - The Eclipse Foundation Open Source Community Website." *Eclipse*. Web. 26 Feb. 2012. <http://www.eclipse.org/>.

25 "Jena – A Semantic Web Framework for Java." *Jena Semantic Web Framework*. Web. 24 Apr. 2011. <http://jena.sourceforge.net/index.html>.

26 "Apache Jena." - *What Is Jena?* Apache Software Foundation. Web. 26 Feb. 2012. <http://incubator.apache.org/jena/about_jena/about.html>.

27 Arendarenko, E., Kakkonen, T.: Ontology-based Information and Event Extraction for Business Intelligence. Submitted for review.

28 "TDB - Jena Wiki." *Jena Semantic Web Framework*. Apache. Web. 27 June 2011. <http://openjena.org/wiki/TDB>.

29 "ARQ - SPARQL Tutorial." *Jena Semantic Web Framework*. Apache. Web. 27 June 2011. <http://openjena.org/ARQ/Tutorial/index.html>.