Semantic Enterprise Technologies

Massimo Ruffolo[1,2], Inderbir Sidhu[2], Luigi Guadagno[2]

[1] ICAR-CNR - Institute of High Performance Computing and Networking
of the Italian National Research Council
[2] fourthcodex inc.
University of Calabria, 87036 Arcavacata di Rende (CS), Italy
e-mail: ruffolo@icar.cnr.it
e-mail: {lguadagno,isidhu}@fourthcodex.com
WWW home page: http://www.fourthcodex.com

Abstract. Nowadays enterprises request information technologies that leverage structured and unstructured information for providing a single integrated view of business problems in order to foster better business process management and decision making. The growing interest in semantic technologies is due to the limitation of existing enterprise information technologies to answer these new challenging needs. Semantic Web Technologies (SWT), the current open standard approaches to semantic technologies based on semantic web languages, provide some interesting answers to novel enterprise needs by allowing to use domain knowledge within applications. However, SWT aren't well suited for enterprise domain because of some drawbacks and a lack of compatibility with enterprise-class applications. This paper presents the new Semantic Enterprise Technologies (SET) paradigm founded on the idea of *Semantic Models* that are executable, flexible and agile representation of domain knowledge. Semantic Models are expressed by means of the *Codex Language* obtained combining Disjunctive Logic Programming (Datalog plus disjunction) and Attribute Grammars both extended by object-oriented and two-dimensional capabilities. Semantic Models enable to exploit domain knowledge for managing both structured and unstructured information. Since the Codex Language derives from the database field, it allows SET to provide advanced semantic capabilities well suited for enterprises. Differences and interoperability issue between SET and SWT are briefly discussed in the paper that shows, also the SET Reference Architecture (SETA), an application example and the business value of SET.

1 Introduction

Nowadays enterprise knowledge workers need information technologies that leverage structured and unstructured information for providing a single integrated view of business problems in order to foster better business process management and decision making. They want answers to specific business requirements, not documents and reports, search document base by concepts not simply by keywords, query databases and unstructured information repositories in a uniform way taking into account the meaning of data and information, obtain application integration and interoperability exploiting semantic-aware services.

The growing area of Semantic Technologies [2] could answer these novel needs by providing a new enterprise-class of semantically-enabled business applications. In this scenario semantics means the use of domain knowledge to affect computing by allowing the meaning of associations among information to be known and processed at execution time. Semantic Technologies must be able to:

- Allow computable semantic representations of domain knowledge, and provide reasoning capabilities on it, in order to enable software to do useful tricks such as finding hidden relationships in a complicated web of objects.
- Handle very large-scale knowledge bases containing both structured and unstructured information.
- Automate the capturing of events of events and entities to connect people, places, and events using information in different formats coming from many different sources.
- Assist human monitoring and analysis of situations, workflows, collaboration and communication.
- Facilitate interoperability by exploiting enterprise concepts to link applications, data sources, and services in easily to use composite views, providing real-time interaction, analysis, and decision-support.
- Deliver their capabilities as value added ingredient components that are easy to embed with existing enterprise applications and integration architectures.

These challenges exceed the capabilities and performance capacity of current open standards approaches to semantic technologies based on semantic web languages (i.e. OWL and RDF) [16]. Such approaches have the benefit to create interoperability over the web but they suffer of the following important drawbacks when applied to enterprise domains: (i) they have a lack of compatibility with relational databases, the most widely adopted enterprise information technologies for representing, storing and managing enterprise structured data; (ii) they allow for "connecting the ontological dots" by using predefined ontological-data model that assists to discover and infer new knowledge (when proper reasoners are available). But 'real world' enterprise business practices are not neatly predefined and in fact are more likely to be dynamic, emergent, or even chaotic. In other words, fully articulated knowledge models (ontologies) are not necessary for recognizing relevant facts in the ever-growing knowledge bases, or inferring new useful related information, or ensuring that enough knowledge is available just in time to improve decision-making; (iii) they do not propose mechanisms for handling directly the already available huge amount of unstructured information.

This paper describes the novel *Semantic Enterprise Technologies* (SET) paradigm founded on a the idea of *Semantic Models* (SM) that are executable, flexible and agile representation of domain knowledge (e.g. simple taxonomies equipped with few and simple descriptors, very rich ontologies equipped with complex business rules and descriptors). SM are represented by means of a new language, called *Codex Language* obtained by combining Disjunctive Logic Programming (Datalog plus disjunction) [4,6,9] and attributes grammars both extended by means of object-oriented [3,12] and two-dimensional capabilities [13,14,15]. SM enable to exploit domain knowledge for managing both structured (e.g. relational databases) and unstructured information (e.g. document repositories).

SET overcome the above mentioned SWT limitations by allowing the following fundamental set of features: (i) they basically came from the database world. So they propose query language and reasoning approaches well suited for the relation model. The Codex language in fact, are based on the Closed World Assumption (CWA) and the Unique Name Assumption (UNA) whereas semantic web languages are based on Open World Assumption (OWA) and do not consider UNA. In order to make SET interoperable with SWT a translation approach that takes into account the different semantics of the Codex Language and OWL has been defined. This way SET can be considered complementary to SWT, (ii) the Codex Language allows to represent *Semantic Models* that can be composed of "just enough" taxonomies and rules or of, if required, complex ontologies containing also relationships, constraints, axioms, so the knowledge representation process fits the very dynamic enterprise environments; (iii) they provide powerful unstructured information management mechanisms that allow concepts annotation and extraction from unstructured sources (semantic enterprise metadata acquisition) via a pattern-based approach. So precise semantic information extraction, classification and search, enabling semantic indexing and querying of the enterprise knowledge, are also possible.

In order to make effective technological and business advantages of SET, as already happens for existing enterprise-class information technologies, a reference architecture that constitutes the framework for applying SET features to enterprise domains is required. This paper proposes SETA (the SET reference Architecture) that describes the technologies and architecture enabling the use of Semantics in an enterprise. SETA aims at transforming multiple sources (structured and unstructured) and bits of dynamic information with domain and concept coverage from disparate enterprise systems into useful knowledge that fosters better enterprise performances.

SET have already been applied to contact-center software, CRM applications, asset and content management repositories, news and media delivery services, health care organizations and more. Their distinctive features help to shape the future of new knowledge-powered computing solutions in many different traditional areas like Competitive Intelligence, Document and Content Management, CRM, Text Analytics, Information Extraction. The application of SETs to real cases shows that they can improve value creation capabilities of enterprises allowing the definition and execution of more efficient and effective business processes and decision making.

The remainder of this paper is organized has follows. Section 2 presents the structure of Semantic Models and describes the Codex Language, Section 3 provides a comparison between Codex Language and OWL and drafts the interoperability approach, Section 4 describes the SET reference Architecture, Section 5 sketches an application of SET to health care risk management, Section 6 contains a brief description of the business value of SET and Section 7 concludes the paper.

2 Semantic Models

SET are based on the concept of Semantic Model. A SM is a flexible and agile representation of domain knowledge. Semantic Models can be constituted by either just small pieces of a domain knowledge (e.g. small taxonomies equipped with few

rules) or rich and complex ontologies (obtained, for example, by translating existing ontologies and by adding rules and descriptors) that gives respectively weak or rich and detailed representation of a domain. More formally a SM is a seven-tuple of the form:

$$SM = \langle \mathcal{H}_C, \mathcal{H}_R, \mathcal{O}, \mathcal{T}, \mathcal{A}, \mathcal{M}, \mathcal{D} \rangle$$

where:

- $\mathcal{H}_C$ and $\mathcal{H}_R$ are sets of classes and relations *schemas*. Each schema is constituted by a set of attributes, the type of each attribute is a class. In both $\mathcal{H}_C$ and $\mathcal{H}_R$ are defined partial orders allowing the representation of concepts and relation taxonomies (with multiple inheritance).
- O and T are sets of class and relation *instances* also called *objects* and *tuples*.
- $\mathcal{A}$ is a set of axioms represented by special rules expressing constraints (rules always true) about the represented knowledge.
- $\mathcal{M}$ is a set of *reasoning modules* that are logic programs constituted by a set of (disjunctive) rules that allows to reason about the represented and stored knowledge, so new knowledge not explicitly declared can be inferred.
- $\mathcal{D}$ is a set of *descriptors* (i.e. production rules in an two-dimensional object-oriented attribute grammar) enabling the recognition (within unstructured documents) of class (concept) instances contained in C, so their annotation, extraction and storing is possible.

SM are represented by means of the novel powerful and very expressive *Codex Language* described in the following.

2.1 The Codex Language

The Codex Language brings together the expressiveness of ontology languages and the power of Disjunctive logic rules. The Codex Language combines notions coming from Disjunctive Logic Programming (Datalog plus disjunction) and Attribute Grammars both extended by means of object-oriented and two-dimensional capabilities. The attribute grammars allow one to intuitively express patterns for recognizing instances of the ontology concepts in structured and unstructured data. The language has been defined as such in recognition of the fact that in order to leverage and apply Semantic Models in the enterprise, it is important to find and retrieve appropriate data from all kinds of data sources, such as, schema based structured databases, unstructured documents and semi-structured documents containing implicit structure.

In order to understand the motivation for extending the capabilities of the Codex Language beyond the modeling capabilities of most ontology languages to a language that also makes it possible to describe how to recognize instances of the ontology concepts in data, consider the following example. Imagine a financial analyst tasked with researching some corporation, say, Microsoft. An ontology could describe that a company may be owned by individuals or boards, and the knowledge base could contain the facts that Microsoft is a company, and it is owned by Bill Gates. Besides representing all of the above information, the codex language can also express the rules and patterns for identifying instances of Microsoft Corp. in documents. One possibility is for a document to mention Microsoft as “the company owned by Bill

Gates". The rules for recognizing the concept of company ownership can easily be expressed in the codex language allowing for the identification of the instances of companies owned by Bill Gates or anyone else in structured and unstructured data sets. Hence, it is now possible to determine that when a document mentions "the company owned by Bill Gates", we have really discovered a document talking about Microsoft Corp.

The Codex language supports the typical ontology constructs, such as, *class*, *object* (class instance), *object-identity*, (*multiple*) *inheritence* and *relation*s, *tuple* (relation instance). It also supports powerful reasoning by means of *reasoning modules* that are modular logic programs containing a set of (disjunctive) rules. The language augments these typical ontology modeling constructs with a mechanism that enables the description of patterns and rules over an ontology for identifying meaningful data in any data source. This part of the codex language extends classical attribute grammars by means of two-dimensional and object-oriented capabilities allowing the expression of concept *descriptors*. A descriptor represents a rule that "describes" the means for recognizing instances (objects) of a concept in unstructured documents by means of complex (two-dimensional) composition of other objects or in structured sources (e.g. databases, structured files) by means of ad-hoc queries and reasoning tasks. When a descriptor matches within an unstructured document, the document can be annotated with respect to the related concept, and moreover, an instance of the matching class can be created in the knowledge base. In order to empower unstructured information management, the Codex language can also exploit sophisticated Natural Language Processing capabilities.

In the Codex Language a *class* can be thought of as an aggregation of individuals (objects or class instances) that have the same set of properties (attributes). A class is defined by a name (which is unique) and an ordered list of attributes identifying the properties of its instances. Each attribute is identified by a name and has a type specified as a built-in or user-defined class. For instance, the classes `country` and `person` can be declared as follows:

```
class country (name:string).
class person (name:string, age:integer, nationality:country).
```

The ability to specify user-defined classes as attribute types (`nationality:country`) allows for the description of complex objects, i.e. objects made of other objects recursively (a person could be a parent that is also a person). The language also supports the definition of special classes called *collection classes* that "collect" individuals that belong together because they share some properties. Instances of these classes can either be declared explicitly as in the case of normal classes, or specified by a rule that defines the shared properties in an intensional way.

Objects (*class instances*) are declared by asserting new facts. Objects are unambiguously identified by their object-identifier (*oid*) and belong to a class. An instance for the class `manager` can be declared as follows:

```
bill:manager("Bill", 35, usa, 50000).
mario:manager("Mario", 37, italy, 40000).
```

Here, the strings "Bill" and "Mario" are the values for the attribute *name*; while `bill` and `mario` are the *object-identifier[s] (oid)* of these instances (each instance is identified by a unique oid). Instance arguments can be specified either by object

identifiers (`usa` and `italy`), or by a nested class predicate (complex term) which works like a function.

Relationships among objects are represented by means of *relations*, which, like classes, are defined by a unique name and an ordered list of attributes (with name and type). Relation instances (*tuples*) are specified by asserting a set of facts. For instance, the relation `managed_by`, and a tuple asserting that project `newgen` is managed by `bill` (note that `newgen` and `bill` are OID), can be declared as follows:

```
relation managed_by(proj:project, man:manager).
    managed_by(newgen, bill).
```

The Codex language makes it possible to specify complex rules and constraints over the ontology constructs, merging, in a simple and natural way, the declarative style of logic programming with the navigational style of ontologies. Additionally, the rules and constraints are organized as *reasoning modules*, benefiting from the advantages of modular programming. Eventually, in order to check the consistency of a knowledge base the user can specify global integrity constraints called *axioms*. For example, the following axiom expresses that each project can have only one manager:

```
::- managed_by(proj:A, man:M1), managed_by(proj:A, man:M2), M1 <> M2.
```

A descriptor can be viewed as an object-oriented production rule $p \in \Pi$ in an two-dimensional attribute grammar defined on a formal context free grammar $\mathcal{G} = \langle \Sigma, V_N, A \in V_N, \Pi \rangle$ over the alphabet Σ. In the Codex Language the domain of the attributes is the set of classes declared in the Semantic Model whereas the alphabet Σ is constituted by class names and object identifiers. More formally, a *descriptor* d is a couple $\langle h, b \rangle$ such that $h \rightarrow b$, where h is the *head* of d and b is its *body*. The following example declares the `company` class, along with an instance and a descriptor for this instance:

```
class company (name:string, nationality:country, market:market_area).
    acme: company("Acme Inc.", usa, rocket_skiis).
    <acme> -> <X: hiStr(), matches( X, "[Aa](?:acme|ACME)")>.
```

The descriptor head `<acme>` represents the object (or the class of objects) that the user desires to recognize within unstructured documents. The descriptor body represents the rule "describing" the structure of the objects in the head in terms of regular expressions (or other objects).

As another example, consider the following fragment of the Codex Language representing the extraction of a table containing stock index:

```
collection class italian_stock_market_index_row(
    stockMarketIndex: stock_market_index, [value]){}
        <italian_stock_market_index_row(stockMarketIndex:S, L)> ->
            <X: stock_market_index(name_index:N, italy)> <X: value(V)>{L:=add(L,V)}+.
collection class italian_stock_market_index_table
    ([italian_stock_market_index_row]){}
        <table(L)> -> <X: italian_stock_market_index_row()> {L:=add(L,X)}+ DIRECTION = "vertical".
```

In the example, the collection class `italian_stock_market_index_row` represents table rows composed of the `stock_market_index` and a sequence of numeric `value[s]`; the collection class `italian_stock_market_index_table` will contains the vertical sequence of rows that constitute the table.

The Codex language allows the expression of very complex patterns that utilize the ability to treat any unstructured or semi-structured document as a two-dimensional plane and exploit the full expressive power of semantic models. The above example

in particular shows the ability to focus on complex and very specific information for extraction from unstructured documents, which in this case happens to be a table of stock indexes related to Italian companies only.

It is noteworthy that *descriptor*[s] can be expressed using a visual support and that the instances of concepts matching the descriptors can be extracted and stored in the Knowledge Base. These instances can also be serialized as XML, RDF, etc. to be used for analytical and/or Web (Semantic Web) based applications.

3 The Semantic Enterprise and the Semantic Web

In order to motivate why Semantic Enterprise Technologies paradigm could represent an opportunity for improving current semantic technology capabilities, it is important to compare and contrast the semantic enterprise approach against the semantic web approach. For example, in the closed world of an enterprise, reasoning using closed world and unique name assumptions offers certain advantages over the open world assumption necessary when working with the Web. Another important aspect for consideration is the need for performing semantic reasoning over enterprise data residing in relational databases. It is also important to be able to integrate rules with ontologies in order to maximize the return from the ontology building effort. The benefits possible from the integration of logic programming and OWL have already been described in [1,8,11]. In this section we provide a detailed description of the advantages of the SET paradigm and its interoperability with semantic web technologies.

3.1 Closed World and Unique Name Assumptions

Enterprise databases are founded on Unique Name Assumption (UNA) and Closed World Assumption (CWA). The semantics of these databases are intuitive and familiar to their users. The CWA is known only information explicitly stored in the Knowledge Base, so a CWA-based rules entails false for information not explicitly declared in the Knowledge Base. The UNA assumes that names of information elements stored in the Knowledge Base are unique, so two elements with different names are necessarily distinct. The codex language is based on closed world and unique name assumption. This makes it highly suitable for modeling and reasoning in the enterprise. Conversely, OWL is based on Open World Assumption which is better suited for the Semantic Web.

A Semantic Model can be seen as an extension of the enterprise database. In order to understand the effect the semantic assumptions of a database can have on the behavior of a query over exactly the same data, consider the databases shown in **Fig. 1.** When database (A) is queried for the company that has its headquarters only in the USA, a CWA-based reasoner returns `Intel`, whereas an OWA-based reasoner is unable to answer this query because there is no explicit statement stating that Intel has its headquarters **only** in the US. Database (B) represents the relationship that for each project we can have at most one leader and

that a project leader can lead many projects. The DL representation is the following TBox: `PROJECT MANAGER hasLeader: PROJECT --> MANAGER` and ABox: `PROJECT(p1) PROJECT(p2) MANAGER(John) MANAGER(Chris) hasLeader(p1,John) hasLeader(p2,Chris)`. In order to represent the database constraint that a project may have at most one manager, we impose the following DL constraint: $\top \leq 1$ `hasLeader`. In the case of a relational database, when a user attempts to insert the fact that John is also a leader of the project `p2`, there is a violation of referential integrity, and the action is not allowed by the database. However, if we assert the same fact in a DL ABox as `hasleader(p2,john)`, the system does not complain about any possible violation. When the system is queried for the leader of the project `p2`, the OWA-based reasoner infers that `Chris` and `John` are the project leaders and that `Chris=John` because of the lack of UNA. In order to obtain the same behavior as CWA in this case, the user must explicitly declare that `Chris` $\neq$ `John`. This is obviously counterintuitive for the normal enterprise user.

A)

COMPANY	HEADQUARTERS	CITY ADDRESS
Microsoft	USA	Seattle
FourthCodex	USA	Chicago
Microsoft	Ireland	Dublin
FourthCodex	Italy	Cosenza
Intel	USA	Saint Clara

B)

PROJECT		
ID	NAME	FK_MAN
1	P1	1
2	P2	2

1:1 hasLeader 0:n

MANAGER	
ID	NAME
1	Chris
2	John

Fig. 1. Two simple relational databases

Negative Queries Negative queries are queries for data where the query condition **does not** hold. For example, an airline database may contain all the pairs of airports connected by its flights. The CEO of the airline might want to query for all the airports not connected by its hub airport. An OWA based reasoner would have trouble answering such a query, since the database does not, and for all practical purposes, can not, contain information about every airport that is not serviced. In the absence of asserted or explicitly derived negative information, the OWL based reasoner cannot answer such query. On the other hand, answering such a query is trivial for CWA based systems which correctly assume that if there doesn't exist any record of a flight between two airports, then it is save to assume that there are not flights between those airports. Answering queries containing negative criteria in an intuitive way usually requires some form of closed-world reasoning.

From the above discussions and examples it follows that CWA and UNA are better suited for the enterprise domain because their semantics are more intuitive for the users. Also, the reasoning based on such assumptions produces more useful new information. So an approach to knowledge representation coming directly from the database world is able to preserve the database semantics allowing the user to query and reason on databases in a more natural way.

3.2 Rules and Integrity Constraints

Rules It is important to integrate rules with ontologies in order to fully capture the knowledge of an enterprise or a domain. Enterprises need to realize a return on their

investment in building ontologies, and a such, ontologies must not be perceived as just a documentation technique for describing a domain, but also as a language and system that can capture and execute domain rules expressed over the ontology. This has also been recognized by the Semantic Web community which is actively working on adding rules to the Semantic Web language stack [7]. The Rule Interchange Format (RIF) working group of the World Wide Web Consortium (W3C) is currently working on standardizing such a language. Responding to popular demand, the Semantic Web Rule Language (SWRL) has been proposed. However, as the authors point out, SWRL has been designed as a first-order language and straightforwardly integrated with OWL as a simple extension. For these reasons SWRL is trivially undecidable and does not address nonmonotonic reasoning tasks, such as expressing integrity constraints. The codex language on the other hand has built-in support for rules. These rules can operate on the concepts, relationships and constraints defined in the ontology, as well as any other arbitrary atoms and predicates desired by the user. The logical underpinning for codex is provided by Datalog, providing closed world semantics such as default negation and unique name assumption as well as support for recursive queries. As consequence of being Datalog based, codex also supports all of the semantics of **SQL** easily extending the benefits of semantic technologies to enterprise databases.

Integrity Constraints In OWL, domain and range restrictions constrain the type of objects that can be related by a role. Also, participation restrictions specify that certain objects have relationships to other objects.

For example, we can state that each student *must* have a student number as: $Student \subseteq \exists hasStudentNumber.StudentNumber$ [10]. However, even though this restriction can be expressed, its semantics are quite different from those of an equivalent constraint in a relational database. A relational database will not allow the user to insert a student without specifying a student number. Because of its open world assumption, OWL will allow a Student without a student number, since the assumption will be that the student has a student number, but it hasn't yet been added to the ABox. Straightforward specification and enforcement of integrity constraints is extremely important for enterprises as such constraints are an essential part of their domain and business model. In the codex language it is trivial to specify and apply such constraints.

3.3 Disjunctive Reasoning

The Codex Language is developed on top of the DLV system [4,9] that allows to exploit the answer set semantics and stable model semantics [6] for disjunctive logic programs. The possibility to exploit disjunction (in the head of rules) and constraints enable to express reasoning tasks for solving complex real problems. The disjunction allows to compute a set of models (search space) that can contain the possible solution for a given problem, whereas constraints allow to choice the solution by adopting a brave or cautious reasoning approach. In the brave reasoning are considered the solutions that are true in at least one model, in the cautious reasoning a solution to be considered acceptable must be true in all the models. Disjunction allows to express very rich business rules and to model reasoning task able to solve different kinds of

problems like: planning problems (under incomplete knowledge), constraints satisfiability, abductive diagnosis. In the following, in order to better explain disjunctive capabilities of the Codex Language an example of team building is provided.

```
module(teamBuilding){
     (r) inTeam(E, P) ∨ outTeam(E, P) :- E:employee(),P:project().
     (c1) :- P:project(numEmp:N),not #count{E:inTeam(E, P)}=N.
     (c2) :- P:project(numSk:S),not #count{Sk:E:employee(skill:Sk), inTeam(E, P)}  ≥ S.
     (c3) :- P:project(budget:B),not #sum{Sa,E:E:employee(salary:Sa), inTeam(E, P)} ≤ B.
     (c4) :- P:project(maxSal:M),not #max{Sa:E:employee(salary:Sa), inTeam(E, P)} ≤ M.  }
```

The reasoning module contain a disjunctive rule `r` that guesses whether an employee is included in the team or not, generating the search space by exploiting the answer set semantics. The constraints `c1`, `c2`, `c3`, and `c4` model the project requirements, filtering out those solutions that do not satisfy the constraints. So knowledge encoded into the Semantic Model can be exploited for providing solutions to complex business problems.

3.4 Interoperability With the Semantic Web

Currently OWL is the standard language on which the Semantic Web movements is trying to really implement it [16]. A lot of dictionaries, thesaurus and ontologies, expressed by means of this language, are already available. Many companies and organizations have invested in the construction of semantic resource like enterprise models and domain ontologies that they want use for building semantic applications. So all the organizations deal with the problem to reuse these semantic resources, developed by means of OWL, and to make them interoperable with already existing databases and application.

The SET paradigm address the interoperability problem with SWT providing an import-export approach that enable to "translate" OWL ontologies in Semantic Models (without descriptors) and viceversa. When a Semantic Model is obtained from an already existing OWL ontologies descriptors can be added in order to enable SET to exploit the model for semantic applications.

To make OWL and the Codex Language interoperable is a complex problem because it requires the translation from OWL (based on description logic) to the Codex Language (based on Disjunctive Logic Programming). Problems related to the joining of DL and DLP has been addressed by many authors [5,8,10]. They presented many methods to translate OWL-DL to logic programming. All these methods require first the definition of which fragment of OWL to deal with. For example, In [10] a detailed description of how a considerable fragment of OWL-DL can be processed within logic programming systems. To this end, the author derives an enhanced characterization of Horn-SHIQ, the description logic for which this translation is possible, and explained how the generated Datalog programs can be used in a standard logic programming paradigm without sacrificing soundness or completeness of reasoning. For the translation from OWL to Codex Language some fragments of the languages for which the semantic equivalence between the original OWL ontology and the obtained Semantic Model is guaranteed have been identified. In this context semantic equivalence means the possibility to obtain the same entailment in the source and in the destination language despite the differences of semantic assumptions. A more detailed discussion

around OWL-Codex Language translation is out of the scope of this paper, for further details is possible to consult [3].

4 SETA: The SET reference Architecture

The SET reference Architecture describes the technologies and architecture enabling the use of Semantics in an enterprise. The various entities for enabling semantic applications in an enterprise are shown in Fig. 2. below.

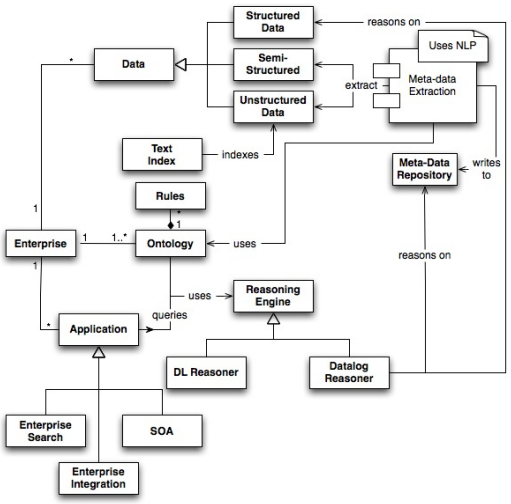

Fig. 2. Semantic Enterprise

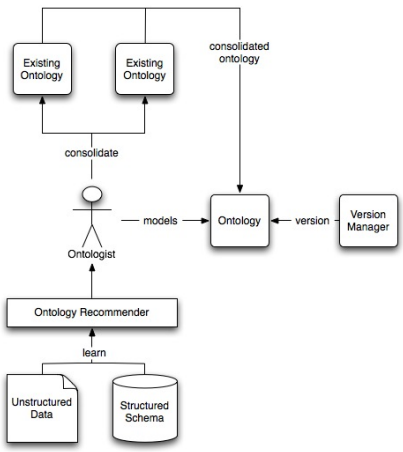

Fig. 3. Ontology Management

In this figure we identify the high level components composing the Semantic Enterprise. These include tools and technologies for:

- modeling and managing Semantic Models
- specifying rules over a Semantic Model
- extracting metadata from unstructured and semi-structured sources
- storing and indexing the extracted metadata
- reasoning over extracted metadata and existing structured data

The modeling environment usually includes a graphical interface (GUI) for creating, modifying and managing Semantic Models. This interface also allows one to specify rules over the Semantic Model. These rules may include the concept descriptors and the reasoning engine uses the constraints and relationships in conjunction with these rules for performing its reasoning tasks. It is vital to include both structured and unstructured data in order to enable the semantic enterprise. This requires two important features from the architecture: (i) use of existing structured data sources, (ii) extraction of useful information from unstructured data sources. The technologies comprising the semantic architecture should be able to use the existing enterprise databases for providing semantic capabilities since the enterprises have large amounts of data and hence it is not feasible to convert all this data into a different format for the purpose of enabling semantic enterprise. So far all of the technologies in the

enterprise have only been able to use structured data for providing decision making capabilities and enabling various applications. With the availability of ontologies and reasoning capabilities it is now possible to leverage unstructured and semi-structured data, since the semantics allow us to assign the correct interpretation. It is important to remember that a large amount of information is locked up in such unstructured sources, and it is important to include it in any decision making process and applications. Extracting useful information from unstructured data requires a multi-pronged approach. This kind of approach includes:

- Natural Language Processing (NLP) including Part of Speech (POS) tagging, sentence splitting, entity extraction, and other NLP related capabilities.
- Concept recognition and extraction.
- Supervised and unsupervised classification.

Supporting the Semantic Enterprise is a well defined process that requires the management of Semantic Models. This goes beyond ontology modeling, and includes the entire ontology life-cycle management comprising of ontology versioning, combination of ontologies into a higher level ontology, ontology comparison, and the addition of rules and descriptors to ontologies as shown in Fig. 3. above.

5 An e-Health Application

In this section an application of SET to a real case is shown. The scope of the application is to support wards to monitor errors and risks causes in lung cancer cares. The application has been developed in the context of a project aimed at provide some Italian hospitals with health care risk management capabilities.

The application takes as input Electronic Medical Records (EMRs) and risk reports coming from different hospital wards. An EMR is generally a flat text document (having usually 3 pages) written in Italian natural language. EMRs are weakly structured, for example, the personal data of the patient are in the top of the document, clinical events (e.g medical exams, surgical operations, diagnosis, etc.) are introduced by a date. Risk reports, filled at the end of clinical process, are provided to patients by wards to acquire information about errors with or without serious outcomes, adverse events, near misses.

The goal of the application is to extract semantic metadata about oncology therapies and errors with temporal data. The application extracts personal information (name, age, address), diagnosis data (diagnosis time, kind of tumor, body part affected by the cancer, cancer progression level), care and therapies information. Extracted information are exploited to construct, for each cared patient, an instance of lung cancer clinical process. Acquired process instances are analyzed by means of data and process mining techniques in order to discover if errors happen following patterns in phases of drugs prescription, preparation or administration.

The application has been obtained by representing a medical Semantic Model inherent to lung cancer that contains: (i) concepts and relationships referred to the disease, its diagnosis, cares in term of surgical operations and chemotherapies with the associated side effects. Concepts related to persons (patients), body parts and risk causes are also represented. All the concepts related to the cancer come from the

ICD9-CM diseases classification system, whereas the chemotherapy drugs taxonomy, is inspired at the Anatomic Therapeutic Chemical (ATC) classification system. (ii) a set of descriptors enabling the automatic acquisition of the above mentioned concepts from Electronic Medical Records (EMRs). In the following a piece of the medical Semantic Model that describes (and allows to extract) patient name, surname, age and disease is shown.

```
class anatomy ().
    class body_part (bp:string) isa {anatomy}.
        class organ isa {body_part}.
            lung: organ("Lung").
            <lung>-><X:histr(), matches(X,"[Ll]ung")>.
        ...
    ...
class disease (name:string).
    tumor: disease("Tumor").
    <tumor>-><X:histr(), matches(X,"[Tt]umor")>.
    cancer: disease("Cancer").
    <cancer>-><X:histr(), matches(X,"[Cc]ancer")>.
    ...
relation synonym (d1:disease,d2:disease)
    synonym(cancer,tumor).
    ...
class body_part_desease () isa {disease}.
    lung_cancer: body_part_disease("Lung cancer").
    <lung_cancer>-><diagnosis_section> CONTAIN <lung> & <X:desease(),synonym(cancer,X)>
    ...
collection class patient_data (){}
    collection class patient_name (name:string){}
        <patient_name(Y)> -> <X:hiStr(),matches(X,"name:")> <X:hiToken()> {Y := X;}
            SEPBY <X:space()>.
    collection class patient_surname (surname:string){}
            <patient_surname(Y)>->
                <X:hiStr(),matches(X,"sur(:?name)?:")> <X:hiToken()> {Y:=X;} SEPBY <X:space()>.
        collection class patient_age (age:integer){}
            <patient_age(Y)>-><X:hiStr(),matches(X,"age:")> <Z:hiToken()>{Y := $str2int(Z);}
                SEPBY <X:space()>.
    ...
collection class patient_data (name:string, surname:string,
                               age:integer, diagnosis:body_part_disease){}
        <patient_data(X,Y,Z,lung_cancer)> ->
            <hospitalization_section> CONTAIN <P:patient_name(X1)>{X:=X1}
            & <P:patient_surname(Y1)>{Y:=Y1} & <P:patient_age(Z1)>{Z:=Z1} & <lung_cancer>.
...
```

The classes `diagnosis_section` and `hospitalization_section` used in the above descriptors represent text paragraphs containing personal data and diagnosis data recognized by proper descriptors that aren't shown for lack of space. The extraction mechanism can be considered in a WOXM fashion: Write Once eXtract Many, in fact the same descriptors can be used to enable the extraction of metadata related to patient affected by `lung_cancer` in unstructured EMRs that have different arrangement. Moreover, descriptors are obtained by automatic writing methods (as happens, for example, for the cancer and tumor concepts) or by visual composition (as happens for `patient_data`)

Metadata extracted by using the Semantic Model are stored as collection class instances into a knowledge base. For the simple piece of Semantic Model shown above the extraction process generates the following `patient_data` class instance for an EMR: `"@1": patient_data("Mario","Rossi","70",lung_cancer)`.

The application is able to process many EMRs and risk reports in a single execution and to store extracted metadata in XML format.

6 The Business Value of Semantic Enterprise Technologies

The SET paradigm allows value creation from different perspectives. From the technological point of view, the value creation capabilities of the SET paradigm can be explained by introducing the *knowledge powered computing* vision. This vision is founded on the transformation of enterprise information into knowledge by converting knowledge into software via a portfolio of embeddable semantic components. Semantic Models transform information into knowledge and can be leveraged to directly embed knowledge into software making it actionable. This way Semantic-aware enterprise applications can be obtained. In fact, to provide value SET cannot be separate, external, or isolated, rather they have to interoperate with the complex portfolios of applications and information repositories, owned by enterprises to enhance their performances.

From the strategic point of view, turning knowledge into software enable to deliver a new generation of *knowledge powered features* that allow to better assist in driving business processes and in making decisions. A new categories of knowledge worker can leverage such enhanced features to obtain functionalities and domain knowledge interchanges that creates information intelligence where knowledge powered applications deliver more precise answers with adaptive responses. So better performances can be achieved by improving decision support and making, exception handling, emergency response, compliance, risk management, situation assessment, command and control.

From the tactical point of view, is important to note that SET features enable a new way to use data and information. SET allow to leverage new information resources like e-mail, web pages, forums, blogs, wikis, CRM transcripts, search logs, organizational documents as already happens for database. To exploit executable Semantic Models enables a better understanding of the interrelationships and shared context of existing structured and unstructured enterprise information. So more informed users can work smarter with better business process execution and monitoring, decision-making and planning.

7 Conclusion

This paper presented the Semantic Enterprise Technologies (SET) paradigm. SET are based on the concept of Semantic Model that are executable, flexible and agile representation of domain knowledge (e.g. simple taxonomies equipped with few and simple descriptors, very rich ontologies equipped with complex business rules and descriptors) and to exploit it for managing both structured (e.g. relation databases) and unstructured information (e.g. document repositories). Semantic Models are expressed by means of the *Codex Language* obtained combining Disjunctive Logic Programming (Datalog plus disjunction) and Attribute Grammars both extended by object-oriented and two-dimensional capabilities. SET overcome the limitation of Semantic Web Technologies (SWT) in enterprise domains. SET provide mechanisms to address several important modeling problems that frequently happens in enterprises and that are hard, if not impossible to solve using OWL alone, but can easily be addressed using the Codex Language. SET are interoperable with SWT thank to a translation

mechanism. Translation allows to import portions of already existing OWL ontologies to use in semantic enterprise applications and to export portions of Semantic Models toward semantic Web applications. Leveraging enhanced semantic features of SET, enterprises can transform information into knowledge in order to achieve better business performances.

References

1. Baader F., Calvanese D., McGuinness D.L., Nardi D., Patel-Schneider P.F., eds. The Description Logic Handbook: Theory, Implementation, and Applications. CUP - 2003
2. Davies J., Studer R., Warren P. Semantic Web Technologies: Trends and Research in Ontology-based Systems. Wiley, July 11, 2006, ISBN-13: 978-0470025963.
3. Dell'Armi T., Gallucci L., Leone N., Ricca F., Schindlauer R. "OntoDLV: an ASP-based System for Enterprise Ontologies". *Proceedings of the 4th International Workshop on Answer Set Programming*, Porto, Portugal, September 8–13, 2007.
4. Eiter T., Gottlob G., Mannila H. Disjunctive Datalog. ACM TODS 22(3) (1997) 364418.
5. Eiter T., Lukasiewicz T., Schindlauer R., Tompits H. Combining Answer Set Programming with Description Logics for the Semantic Web. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada. (2004) 141–151.
6. Gelfond M., Lifschitz V. Classical Negation in Logic Programs and Disjunctive Databases. NGC, vol. 9, pg. 365-385, 1991.
7. Horrocks I., Patel-Schneider P.F., Boley H., Tabet S., Grosof B., Dean M. Swrl: A semantic web rule language combining owl and ruleml W3C Member (2004) Submission. http://www.w3.org/Submission/SWRL/.
8. Krotzsch M., Hitzler P., Vrandecic D., Sintek M. How to reason with OWL in a logic programming system. In Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web, RuleML2006, pp. 17–26. IEEE Computer Society, Athens, Georgia, November 2006.
9. Leone N., Pfeifer G., Faber W., Eiter T., Gottlob G., Perri S., Scarcello F. The DLV System for Knowledge Representation and Reasoning. ACM TOCL 7(3) (2006) 499562
10. Motik B. Reasoning in Description Logics using Resolution and Deductive Databases. Phd Thesis. 2006 Karlsruhe.
11. Motik B., Horrocks I., Rosati R., Sattler U. Can OWL and Logic Programming Live Together Happily Ever After? 5th International Semantic Web Conference, Athens, GA, USA, November 5-9, 2006, LNCS 4273.
12. Ricca F., Leone N. "Disjunctive Logic Programming with types and objects: The DLV+ System". *Journal of Applied Logic*, Elsevier, Volume 5, Issue 3, September 2007, Pages 545-573.
13. Ruffolo M., Leone N., Manna M., Sacc D., Zavatto A. Esploiting ASP for Semantic Information Extraction. In proceedings of the ASP'05 workshop - Answer Set Programming: Advances in Theory and Implementation, University of Bath, Bath, UK, 27th–29th July 2005.
14. Ruffolo M., Manna M. A Logic-Based Approach to Semantic Information Extraction. In proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS'06), Paphos, Cyprus, May 23-27, 2006
15. Ruffolo M., Manna M., Oro E. Object Grammar. Internal Report of High Performance Computing and Networking Institute of the Italian National Research Council. 2007
16. Smith M. K., Welty C., McGuinness D. L. OWL web ontology language guide. W3C Candidate Recommendation (2003) http://www.w3.org/TR/owl-guide/.