Agile development of ontologies through conversation

Dave Braines [*a], Amardeep Bhattal [a], Alun Preece [b], Geeth de Mel [c]
[a] Emerging Technology Services, IBM United Kingdom Ltd, Hursley Park, Winchester, UK
[b] School of Computer Science and Informatics, Cardiff University, Cardiff, UK
[c] IBM Research UK, STFC Daresbury Laboratory, Daresbury, Warrington, UK

ABSTRACT

Ontologies and semantic systems are necessarily complex but offer great potential in terms of their ability to fuse information from multiple sources in support of situation awareness. Current approaches do not place the ontologies directly into the hands of the end user in the field but instead hide them away behind traditional applications. We have been experimenting with human-friendly ontologies and conversational interactions to enable non-technical business users to interact with and extend these dynamically. In this paper we outline our approach via a worked example, covering: OWL ontologies, ITA Controlled English, Sensor/mission matching and conversational interactions between human and machine agents.

Keywords: Ontologies, Controlled Natural Language, Conversational Interaction, Sensor / Mission Matching, OWL

1. INTRODUCTION

Ontologies and, more generally, semantic representations offer great potential for knowledge capture and unambiguous information representation, especially when it comes to support from machine agents in the form of reasoning and algorithm execution. In traditional systems ontologies can be complex to develop and understand and are usually the focus of technical specialists and knowledge engineers rather that business-focused "end users". In our previous work we have experimented with Controlled Natural Languages[1], specifically ITA Controlled English (CE)[2] in an attempt to present ontological information in a more human-friendly, consumable form, aimed directly at non-technical business users. In our most recent work we have demonstrated a full natural language conversational capability[3] that enables human users to interact with a Controlled Natural Language Knowledge Base in their own natural language (e.g. English) in order to assert new knowledge, explore the current knowledge base (both in terms of the "model" and the "facts") and direct their questions to specific human or machine agents[3]. Our initial experiments with human subjects demonstrate that untrained users are able to assert basic knowledge into the knowledge base and ask questions in both solitary (offline) and cooperative group (online) settings[4].

In this paper we outline the potential for extending the knowledge base model (a.k.a the ontology[5]) via these natural language conversational speech acts and give some brief worked examples to show how this would be experienced along with a detailed explanation and demonstration of how such information when authored in a CE knowledge base can be easily converted into more traditional ontological formalisms using OWL[6] (the Semantic Web Ontology Language). The main purpose of this paper is to provide a detailed exposition of how to automatically generate OWL ontologies from simple CE models, enabling the reader to build far more complex models in the CE language and generate the require OWL ontologies. To demonstrate the scalability of the approach the results of a simple experiment to generate an OWL ontology for a complex sensor-mission matching ontology are given towards the end of the paper.

The remainder of this paper is structured as follows: Section 2 gives a brief outline of the anticipated conversational basis for gathering ontology extensions from business users (e.g. personnel in the field) including mechanisms for representation and implementation of such changes. Section 3 gives a detailed description of a mechanism for the

[*] Send correspondence to Dave Braines, E-mail: dave_braines@uk.ibm.com, Tel: +44 (0) 1962 818749

conversion of such information from a CE knowledge base into OWL ontologies. The paper is concluded in section 4 with a brief discussion of related work and potential further activities.

2. ONTOLOGY EXTENSION THROUGH CONVERSATION

In earlier work[3] we developed a simple model of conversational interaction based on speech act theory and embodied through the creation of a number of "card" concepts with different specializations which enable conversations to flow between human and machine agents. The cards themselves are expressed as CE instances and they contain a "payload" which is the content of the conversational act as well as a rich set of meta-data including the date/time of the speech act, the card which it is in reply to, the user (human or machine) which created it and the user(s) (human or machine) to which it is sent. The "payload" (or "content" in terms of the attribute name within the CE model) is the text which is the utterance made by the agent in question. Depending on the card type this can either be Natural Language (NL) or Controlled English (CE), with a rich set of specialisations for CE cards to ensure that the context intended for the CE card is captured correctly. For NL cards the context must be inferred based on a number of contextual factors such as the NL text itself, the position in the dialogue, the users sending or receiving the message etc. These cards are generated by both human and machine agents, can contain either NL or CE and support human-human, human-machine, machine-human and machine-machine communications.

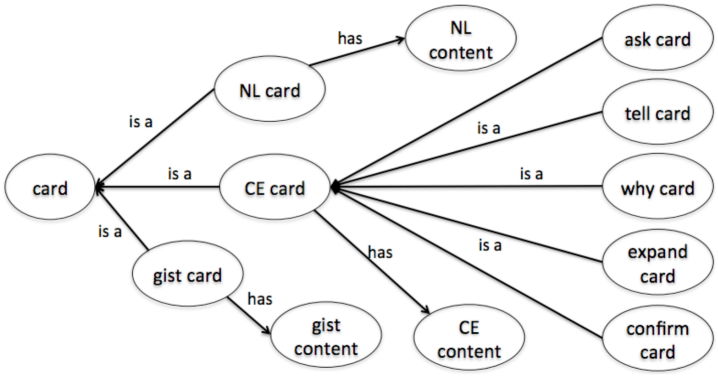

Figure 1. The "CE card" conceptual model

We also developed a protocol[3], again based on speech act theory, identifying which acts are commonly encountered together and therefore the basic building blocks from which a multi-turn dialogue based conversation can be built. Using these simple but powerful definition we have since built a number of demonstration applications and performed a series of simple experiments to determine whether untrained human users are able to use this mechanism when embodied in a simple messenger style chat interface to successfully communicate with machine agents and collaboratively contribute new knowledge to an existing ontology in both solitary and collaborative modes of operation[4].

2.1 Extending ontologies through conversation

One of the guiding principles of the CE language is that it should be aimed at the business user and should use the language of their domain of interest rather than forcing them to use ontological terms and set-theoretic expressions[2]. This is motivated from the desire to appeal directly to non-technical users rather than technical specialists and knowledge engineers and in this paper we extend this principle to ontology extension as well, specifically in the assertion that many "simple" ontology extensions could provide significant benefit to a human/machine hybrid system. For example the addition of a new concept (class) that provides a meaningful specialization of one or more existing concepts, or the addition of new relationships or properties, the definition of new simple rules etc. All of these could support field users in their attempts to define an understanding of the world, especially in highly agile situations such as disaster relief or incident support. This does not rule out the more complex ontology extensions that could be made, nor does it rule out user-defined meta-model extensions that an advanced CE user can make to extend the capabilities of the CE language itself. These are not documented in this paper, however they would be implemented in the same simple mechanism, but using the language of the meta-model. For example if a user had chosen to add the concept of "disjointness" to the CE meta-model (it is not enforced for all CE users since it is not a concept that is critical form domain modeling) then new

disjoint assertions could still be captured via conversation since the concept of disjointness would be available within the CE model and therefore a potential topic of discussion along with all of the users domain terminology.

Therefore, this section does not attempt to provide a definitive list of all possible ontological extensions but instead provides a single narrative using a simple worked example based on a disaster relief scenario involving a flooding situation and the need for rapid deployment of food and medical supplies to assist the local population in recovery from the flood. The scenario is motivated by the desire to reuse existing publically available information to maximize the productivity of the field team and ensure that the situation is progressed as quickly as possible. In this scenario the system has access to data on UK schools, based on data made publically available via the UK government[7] and it is assumed that this data is already available in the CE knowledge base. Earlier work has demonstrated that the existing csv data can easily be described and subsequently translated into the CE language based on a simple csv conversion process[8]†.

Conversation 1: Creating a new concept

In this conversation we show how the human user creates a new concept within the knowledge base.

Field user: *I need to identify possible staging areas*

Machine agent: *I don't understand*

This initial dialogue is intentionally included to remind the reader that we are using a simple "bag of words" parsing system so it is unrealistic to expect that the machine agents will be able to understand all forms of Natural Language, especially in cases such as this where some of the given words are the name to be given to a new feature within the knowledge base. With further configuration and tuning the system could be developed further to correctly interpret a wider range of input patterns, however it will always remain a simple mechanism. So let's try again:

Field user: *Create new concept named possible staging area*

Machine agent: *Did you mean:*
`conceptualise a ~ possible staging area ~ P.`
?

Field user: *Yes*

Machine agent: *I have added that knowledge to the knowledge base.*

In this second attempt we show the interaction that successfully creates a new concept within the knowledge base and confirms the creation back to the user. The text in `this style` is valid CE and in this example is shown directly to the user for their interpretation and confirmation. This is because in this scenario we envisage that the field user has some exposure to the system and some familiarity with the CE language, however if this was a novice/casual user then we could employ the "gist" style card to transform the raw CE into a natural language summary that is more immediately consumable to casual users but still demonstrates that the machine agent has correctly interpreted the natural language text provided by the human field user.

Conversation 2: Identifying instances

The field user now wishes to query information from the CE knowledge base, in this case any known information about existing "sites".

Field user: *List known sites nearby*

---

† There is also an existing video (here – https://www.usukitacs.com/sites/default/files/UKdata.mp4) that demonstrates the definition of the "possible staging area" concept against the school data, but this is in pure CE since it pre-dates the most recent conversational interaction research.

Machine agent: *I am aware of schools – there are 7 within a 10km radius*

Through concepts defined in the ontology (and potentially the use of synonyms if the human field user wishes to use different terminology to the defined ontology) the system is able to interpret this request and identify that the field user is asking about sites. In the knowledge base the concept "school" is defined as being a sub-concept of the "site" concept and therefore the machine agent is able to provide a meaningful response. Part of the meta-data provided in the CE cards model is the geo-location of the users that are generating the cards, and this knowledge in addition to the lat/lon coordinates provided with the schools data enable the machine agent to reply with a contextually meaningful answer, identifying how many instances are "nearby". The concept of nearby (e.g. 10km or 1km or some other value) is also able to be defined in the CE knowledge base, and the human user can override this default assumption if needed, e.g. if they are asking the question ahead of moving to some other area of operation.

Field user: *Show details*

Machine agent: *(lists all attributes for the 7 schools)*

In this interaction the machine agent applies the knowledge of the previous dialogue to understand that "*show details*" is to be taken in the context of the previous speech act and is able to show the details of the 7 school instances directly to the user. As previously shown, this can either be in CE directly, or in a summarized natural language "gist" form, depending on the configuration of the system. In our example we are basing the knowledge of school attributes purely on the data available from the public UK government data. This does not contain any information about building size or location, and does not contain elevation data (which would be of critical importance in a flooding situation). It is likely that elevation data for a given latitude and longitude coordinate could be provided by a separate generic service and if available the school data could be automatically augmented with this data by a machine agent invoking the service and adding the elevation data to the CE knowledge base before returning the result to the user, but that is ignored for this simple scenario.

Upon seeing the school data the field user is initially disappointed that more meaningful/relevant information is not available, but they then realize that they can make a "quick and dirty" assumption based on their own experiences of the UK school system. The "common sense" rule that comes into the field users head is "A school will generally have a big hall if it has a large number of students" – because schools tend to gather their pupils together for assembly meetings. Since the field user is seeking to identify possible staging areas they know they'll need locations which have large internal spaces that could shelter displaced people or provide storage for supplies and equipment. The field user realizes that they can use this knowledge to rule out certain schools and automatically classify others as possible staging areas:

Field user: *If a school has more than 1000 students then it is a possible staging area.*

Machine agent: *Did you mean:*

```
if
  ( the school S has the value N as number of students ) and
  ( the value N > 1000 )
then
  ( the school S is a possibly staging area ).
?
```

Field user: *Yes*

Machine agent: *I have added that knowledge to the knowledge base.*

This simple exchange demonstrates the definition of a formal semantic rule (in CE) as a result of the realization of the field user that their "common sense rule" could be useful. They could also have done the same for a flooding rule if elevation data were available (although clearly there could be access issues if the school in question were locally elevated and a hill, but cut off from all access roads). The confirmation of the rule by the field user asserts the rule into

the knowledge base at which point it can be executed by a simple machine agent. This execution leads to the inference of a number of new instances of "possible staging area" based on the existing school data and the recently added rule. As a result of the earlier question by the field user the machine agent is able to determine that these new instances of possible staging area will be of potential interest to the field user and the machine agent interjects with a new alert (in the form of a standard card in the normal conversational interaction) advising the field user of this new knowledge. Such an alert could be sent to many users, not just the user who created the new rule. This would be based on a number of criteria such as proximity, role and previous discussions.

Machine agent: *Be aware, there are 3 possible staging areas in your vicinity*

There are many more simple conversational acts that could be demonstrated to show the potential power of a knowledge base system that is directly accessible to, and extensible by, non-technical field users, but we do not have the space to provide these examples in this paper. The remainder of the paper provides detail on how CE knowledge bases can be converted into OWL ontologies to allow integration from this potentially powerful CE form that is aimed at provide agile capabilities to edge users without specialist knowledge of ontology languages. The ability to seamlessly transform to and from OWL ontologies offers the potential for integration into the wider Semantic Web community and the many approaches that make use of the Open Standard OWL approach.

3. OWL ONTOLOGIES FROM ITA CONTROLLED ENGLISH

A knowledge base system models a domain of interest and allows reasoning over data populating the domain. One approach to producing such a system is to use the Web Ontology Language (OWL) and an associated software 'reasoner'. However, whilst domain experts can express the entities and their behaviours required for such a system in an abstract sense, converting this into OWL requires detailed knowledge of the semantics of OWL, as well as being comfortable with the chosen syntax or tooling. Furthermore, the most commonly used serialization of OWL (RDF/XML) is not particularly intuitive to a non-expert user.

An alternative approach is to use a controlled natural language (CNL), which enforces precision and a lack of ambiguity in the statements made, but using a readily understandable syntax. The difficulty in using OWL directly, and the possibility of using a CNL was investigated by Schwitter et al[9], who compared three CNLs that were each produced with the aim of authoring OWL, or having this as a significant use-case.

Another approach is to use a CNL to define and interact with the knowledge base, but with an option to export this to an OWL representation if required. The emphasis is then on expressing the CNL in OWL, rather than OWL semantics in a given CNL (with a view to ultimately generating OWL output). Here the conversion of ITA Controlled English (CE) into OWL is investigated.

Translatable CE semantics and features

CE concentrates on a level of expressivity that allows useful knowledge bases to be built, whilst keeping language features to a minimum. The built-in features are confined to value (string) types, and user-defined complex types, which may have properties which are either values, or user-defined complex types. Thus, the built-in features are fairly minimal. However, the language allows richer behaviours to be modelled through the capability to define rules that act on instances of these types, and the ability to interact with the underlying CE metamodel (a model of the base built-in CE concepts that are instantiated to represent the user's knowledge base).

There are three implementations of CE in use within the ITA[10] project, a Java-based implementation known as **ce-store**[‡], a lightweight NodeJS implementation known as **CENode**[§], and a rich Prolog-based implementation known as **ce-prolog**[**]. The CE language features provided by each differ, with the ce-prolog implementation being (approximately) a small superset of the ce-store Java implementation and the CENode implementation being a smaller subset still, designed for use at the very edge of the network within a browser on a mobile device. All three implementations support CE model and CE fact sentences that are described later in this paper.

[‡] ce-store is available as open source, for download at http://github.com/ce-store

[§] CENode is available as open source, for download at http://cenode.io

[**] ce-prolog is not yet publically available

The fuller implementations of the CE language include additional capabilities in the areas of the expressiveness available when interacting with the metamodel (for example the ability to define a general case symmetric property rule rather than multiple rules specific to named types), defining ordered sequences, treating CE statements as variables, and features to assist in handling assumptions and uncertainty. The Java ce-store implementation includes a sample graphical user interface (GUI) known as the "Engineering Panel", HTTP APIs for user code to interact with, and hook points for user 'agent' and 'trigger' code within the Java runtime environment. In this section we concentrate on translating CE held in ce-store to OWL's RDF/XML serialization format.

Mace et al[11] investigated which semantic features of OWL could be expressed in CE. In addition, new syntax to enable the expression of missing OWL semantics in CE was suggested, as well as extending the syntax to provide more natural means to express semantics that were expressible, but in a non-compact or unintuitive form. The work also reinforced the point that CE rules play a significant part in the language. So, although some features such as transitive or symmetric relations are not built-in (unlike in OWL), rules may be defined to provide the required behaviour. For example, a rule that acts on instances of given types may enforce a symmetric relation in a manner similar to the example that follows, *'if the person A is a friend of the person B, then the person B is a friend of the person A*'. A corollary of this investigation was the identification of the features of OWL that are already provided by CE.

Another approach is to examine the OWL metamodel that is defined by[12]. This model describes the built-in types and their relations, instances of which are created to represent a user's ontology. The document remarks that '*The structural specification of OWL 2 consists of all the figures in this document and the notion of structural equivalence given below. It is used throughout this document to precisely specify the structure of OWL 2 ontologies and the observable behaviour of OWL 2 tools. An OWL 2 tool MAY base its APIs and/or internal storage model on the structural specification*'. Thus inspection of the UML figures within the document is a good way to understand the available OWL semantics and how they are related, and so enumerate the available options for mapping from CE to OWL.

Although this is an instructive process, the conclusion from this, and from considering the Mace document, is that the 'obvious' mapping is the simplest and best choice - CE concepts map to OWL classes, CE value properties map to OWL data properties, and CE concept properties map to OWL object properties. Instances of concepts (including any properties) map to OWL individuals of the appropriate class (with the appropriate properties). The richness encoded in CE rules is hard to translate into OWL from a practical point of view, as unlike the type and instance systems, there is no built-in model of these rules available to inspect at run-time, hence it is hard to find what has been encoded into the rules a user has defined. However, even if rules are not translated, the other features will suffice for taxonomies and simpler models.

CE also allows properties to be expressed in two syntactical forms, 'functional-noun' and 'verb-singular'. The distinction between these two alternatives was deemed not worth capturing as the choice is purely linguistic or syntactical and has no effect on the properties defined, and furthermore there is no natural fit in OWL (though if desired, this information could be captured, perhaps by using OWL annotations).

Mapping CE to OWL ontologies

Although the mapping of CE concepts, properties and instances to equivalent OWL constructs is readily achievable, CE and OWL differ in how their statements are partitioned or grouped.

Leaving aside rules and queries, CE has two types of statements, model sentences and fact sentences (the latter of which create instances of the concepts and properties defined in the former). It also has a flat namespace so that all sentences are globally visible. However, any model sentences that define concepts or property concepts may be headed by an annotation declaring a 'model' name. The sentences then belong to the named model, which is purely used for convenience to identify related model sentences. Any model sentences not in the scope of a declared model annotation are placed in a default model named 'global'.

ce-store also uses another mechanism to group sentences. A 'source' is associated with any sentences that are loaded into the system (sources are specific to the ce-store implementation, rather than being a defined part of the CE language). For example, loading a file into the system via a CE command creates a new source for the file's contents, entering CE via the ce-store GUI uses another source, as does uploading a file via the GUI. A source may contain both fact and model sentences (for example it is permissible for a CE file to contain both types of sentence), but a model only contains model sentences.

OWL statements do not inhabit a flat, globally-visible namespace, instead they are grouped into ontologies, with an explicit import being required to make statements in one ontology visible to another ontology. In addition, OWL uses IRIs (internationalised URIs) to identify OWL elements such as classes, properties, instances, and ontologies as a whole. In OWL's RDF/XML serialization, XML namespaces may be introduced to aid the construction of the contained IRIs.

One common approach is to place a single ontology in a file, declare elements using IRIs in an XML namespace specific to the ontology (usually the same XML namespace as that of the ontology IRI), set the xml:base XML element to this namespace to use it by default in any places in the file where absolute IRIs are not used, and import ontologies that declare any referenced external elements (for example, class or property definitions referred to in the declaration of instances, or an external class definition used when creating a subclass).

Note that the XML namespace used in IRIs uses a different notion of the term 'namespace' to that used previously here. An XML namespace is an XML construct in the RDF/XML serialization that aids constructing IRIs. The previous usage of the term namespace was used to describe the mutual visibility (or otherwise) of CE sentences or OWL statements. In OWL this visibility is controlled by placing the statements into ontologies, and then importing other ontologies as desired to make their statements visible.

The simple scheme of using one XML namespace per ontology (which includes all the elements declared within the ontology) is not practical if some element, say a class, has definitions spread across multiple ontologies. An example of this is to declare a class in one ontology, and declare it to be a subclass of some other class in another ontology. Since the class is identified by its IRI, this must be constant in all the ontologies that contribute to the class definition. This requirement means that the simplistic scheme outlined above must be relaxed so that each ontology does not use a unique namespace for all the elements it declares or modifies. Note that one ontology must still import another to see the statements that the second contains, regardless of the XML namespaces used.

We choose to map CE sentences to OWL ontologies such that the sentences in each model are mapped to elements in a corresponding ontology (so that there is an OWL ontology per CE model). In addition, all fact sentences from a given source are mapped to a corresponding ontology (so there is an OWL ontology per source that contains fact sentences). Thus we separate model statements from fact statements, and further partition by model and source. This arrangement avoids the production of one large monolithic ontology containing the entire contents of the ce-store system, aiding legibility and componentising the knowledge base into understandable sub-units.

Due to CE's flat namespace it is trivial to refer to and add to the definitions for a given concept across multiple models - the name of each concept is assumed to be unique (so any reference to the same name must refer to the same concept) and import statements are not required (since all CE sentences are loaded into a single unpartitioned namespace).

Consider loading the following CE into ce-store:

```
Model: one
conceptualise a ~ person ~ P.
```

and then loading the following :

```
Model : two
conceptualise a ~ person ~ P that
        has the value A as ~ age ~.
```

Model one declares a person concept, and the subsequent load of model two could be interpreted as the addition of an age property to the concept. In fact both sentences are complete definitions for a person concept, with the end result being the union of the definitions. In this case the result of the union is the same as just using model two. There is no

explicit way in CE to add a property to some concept that must have been declared elsewhere, when model two is loaded the person concept is effectively declared again with an age property as an additional part of this redeclaration.

There is no single 'root' definition for a concept whose definitions span multiple models. In the case above we could attempt to use specific XML namespaces for each resulting ontology, and use the namespace for model one wherever we refer to the person concept, following our intuition that the 'plain' version of the concept (without any properties) is the root definition. However, in general such a definition may be present in multiple models, or indeed there may be no plain concept declared anywhere (for example the concept may be declared with various properties in various models). The assumption that the plain concept declaration is somehow more significant that others is questionable in any case as explained previously.

In conclusion, concept declarations may easily span models in CE. A common mechanism by which this occurs is to declare properties for a given concept in different models, or to declare a plain concept without any properties in one model and add properties elsewhere. In addition, picking a single 'root' model that defines the concept (with a view to determining a single XML namespace for a class, based on that of a 'root' defining ontology for the class) is also impractical.

The simplest solution to this situation is to mirror CE and use a common XML namespace in all ontologies, and for all the elements declared within the ontology. This means that, as required, a single OWL class is declared for any given CE concept, regardless of the ontologies it spans, and properties also inhabit the same XML namespace.

We can however use separate XML namespaces for model and instance statements since the statements in each of these two partitions do not add to the definitions for elements in the other partition (the only dependency is for the required ontology import statements in instance ontologies for the model declarations they reference but do not modify).

Implementation

In order to evaluate the proposed approach a basic implementation was developed using the simple machine agent infrastructure within the ce-store environment. The agent outputs each ontology to a separate file. The agent concept is defined in the usual manner, together with a named value that will be used to hold the destination directory for the resulting files:

```
conceptualise an ~ owl output agent ~ A that
        is a CE agent and
        has the value V as ~ output directory path ~.
```

An example configuration is:

```
there is an owl output agent named 'ce to owl agent' that
        has 'com.CeToOwlAgent' as class name  and
        has '/Users/myuser/OWLOutput' as output directory path.
```

Full implementation details are not provided here for reasons of brevity, however the full implementation will be subsequently published as a technical report before the completion of the ITA research program. The remainder of this section provides some additional detail about some parts of the implementation.

XML namespace settings

If desired, the XML namespaces used in the generated OWL may be modified by small alterations to the configuration of the agent.

By default the following settings are used,

- URI_ROOT_MODEL = http://www.ita-ce.com/model
- URI_ROOT_INSTANCE = http://www.ita-ce.com/instance

These settings result in the behaviour outlined previously (all model statements in one XML namespace, and all fact statements in another).

To place all model and fact statements in the same XML namespace, edit the code so that,

- URI_ROOT_MODEL = http://www.ita-ce.com
- URI_ROOT_INSTANCE = http://www.ita-ce.com

The two URIs may be set to any identical value.

Mapping multiple dependent CE models to OWL

Given three files containing the following model CE:

```
Model: Super Model

conceptualise a ~ superclass ~ C.
```

```
Model: Super Prop Model

conceptualise a ~ superclass ~ C that
        has the value V as ~ valueproperty ~.
```

```
Model: Sub Model

conceptualise a ~ subclass ~ C that is a superclass.
```

And given a file containing the following fact CE,

```
there is a superclass named 'my superclass'.

there is a subclass named 'my subclass' that
 has the value 'my value' as valueproperty.
```

Then three corresponding OWL ontologies are produced for the three CE models, each in their own files,

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY owl  "http://www.w3.org/2002/07/owl#">]>

<rdf:RDF
  xmlns:xsd="&xsd;"
  xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;"
  xmlns:owl="&owl;"
  xml:base="http://www.ita-ce.com/model">

  <owl:Ontology rdf:about="http://www.ita-ce.com/model/supermodel">
  </owl:Ontology>

  <owl:Class rdf:ID="superclass">
  </owl:Class>

</rdf:RDF>
```

```
  <!-- xml headers and footers removed for brevity, xml:base is as before -->

  <owl:Ontology rdf:about="http://www.ita-ce.com/model/superpropmodel">
    <owl:imports rdf:resource="http://www.ita-ce.com/model/supermodel"/>
  </owl:Ontology>

  <owl:Class rdf:ID="superclass">
  </owl:Class>

  <owl:DatatypeProperty rdf:ID="valueproperty">
    <rdfs:domain rdf:resource="#superclass"/>
    <rdfs:range  rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
```

```
  <!-- xml headers and footers removed for brevity, xml:base is as before -->

  <owl:Ontology rdf:about="http://www.ita-ce.com/model/submodel">
    <owl:imports rdf:resource="http://www.ita-ce.com/model/supermodel"/>
    <owl:imports rdf:resource="http://www.ita-ce.com/model/superpropmodel"/>
  </owl:Ontology>

  <owl:Class rdf:ID="subclass">
    <rdfs:subClassOf rdf:resource="#superclass"/>
  </owl:Class>
```

And a single OWL ontology is produced to correspond to the CE fact sentences associated with the source that was used when loading the CE fact file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#">
 <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
 <!ENTITY owl  "http://www.w3.org/2002/07/owl#">
 <!ENTITY submodel "http://www.ita-ce.com/model/submodel#">
 <!ENTITY supermodel "http://www.ita-ce.com/model/supermodel#">
 <!ENTITY superpropmodel "http://www.ita-ce.com/model/superpropmodel#">
 <!ENTITY model "http://www.ita-ce.com/model#">]>

<rdf:RDF
 xmlns:xsd="&xsd;"
 xmlns:rdf="&rdf;"
 xmlns:rdfs="&rdfs;"
 xmlns:owl="&owl;"
 xmlns:nssubmodel="&submodel;"
 xmlns:nssupermodel="&supermodel;"
 xmlns:nssuperpropmodel="&superpropmodel;"
 xmlns:nsmodel="&model;"
 xml:base="http://www.ita-ce.com/instance">

<owl:Ontology rdf:about="http://www.ita-ce.com/instance/src010">
   <owl:imports rdf:resource="http://www.ita-ce.com/model/submodel"/>
   <owl:imports rdf:resource="http://www.ita-ce.com/model/supermodel"/>
   <owl:imports rdf:resource="http://www.ita-ce.com/model/superpropmodel"/>
 </owl:Ontology>

 <nsmodel:subclass rdf:ID="mysubclass">
   <nsmodel:valueproperty rdf:datatype="&xsd;string">my value</nsmodel:valueproperty>
 </nsmodel:subclass>

 <nsmodel:superclass rdf:ID="mysuperclass">
 </nsmodel:superclass>

</rdf:RDF>
```

Mapping a CE instance (with multiple CE concepts)

Given the following fact CE sentences (and appropriate model CE to match):

```
there is a concept 1 named 'multiple concept instance'.

the concept 1 'multiple concept instance' is a concept 2.
```

Then the generated corresponding OWL is,

```
<owl:Thing rdf:ID="multipleconceptinstance">
  <rdf:type rdf:resource="&model;concept1"/>
  <rdf:type rdf:resource="&model;concept2"/>
</owl:Thing>
```

Note that here we use a slightly different form of OWL to declare the types for the instance to that used when a single class is involved.

Visualising translated OWL

The previous examples given in this document use concise, simple CE examples to convey the way that CE is converted into corresponding OWL elements. This example proves that an existing, large and realistically rich CE model can also be successfully translated. The example used is the ISTAR asset model as described in "Tasking and Sharing Sensing Assets Using Controlled Natural Language"[13]. The model describes sensors and capabilities for intelligence, surveillance and reconnaissance scenarios. The CE version of this model converts into an OWL file that loads without error into Protégé[14], which in turn reports that the ontology contains 115 classes and 18 properties. It also successfully loads into the WebVOWL online OWL visualisation tool[15], which produced the image of the ontology shown below, giving an impression of the overall hierarchy and organisation of the classes within the ontology. Please note that the image is not intended to be readable at this scale but is included here to show the size and complexity of the existing complex CE model that was successfully converted to OWL without modification, such that existing OWL tooling is able to make immediate use of the generated ontology information.

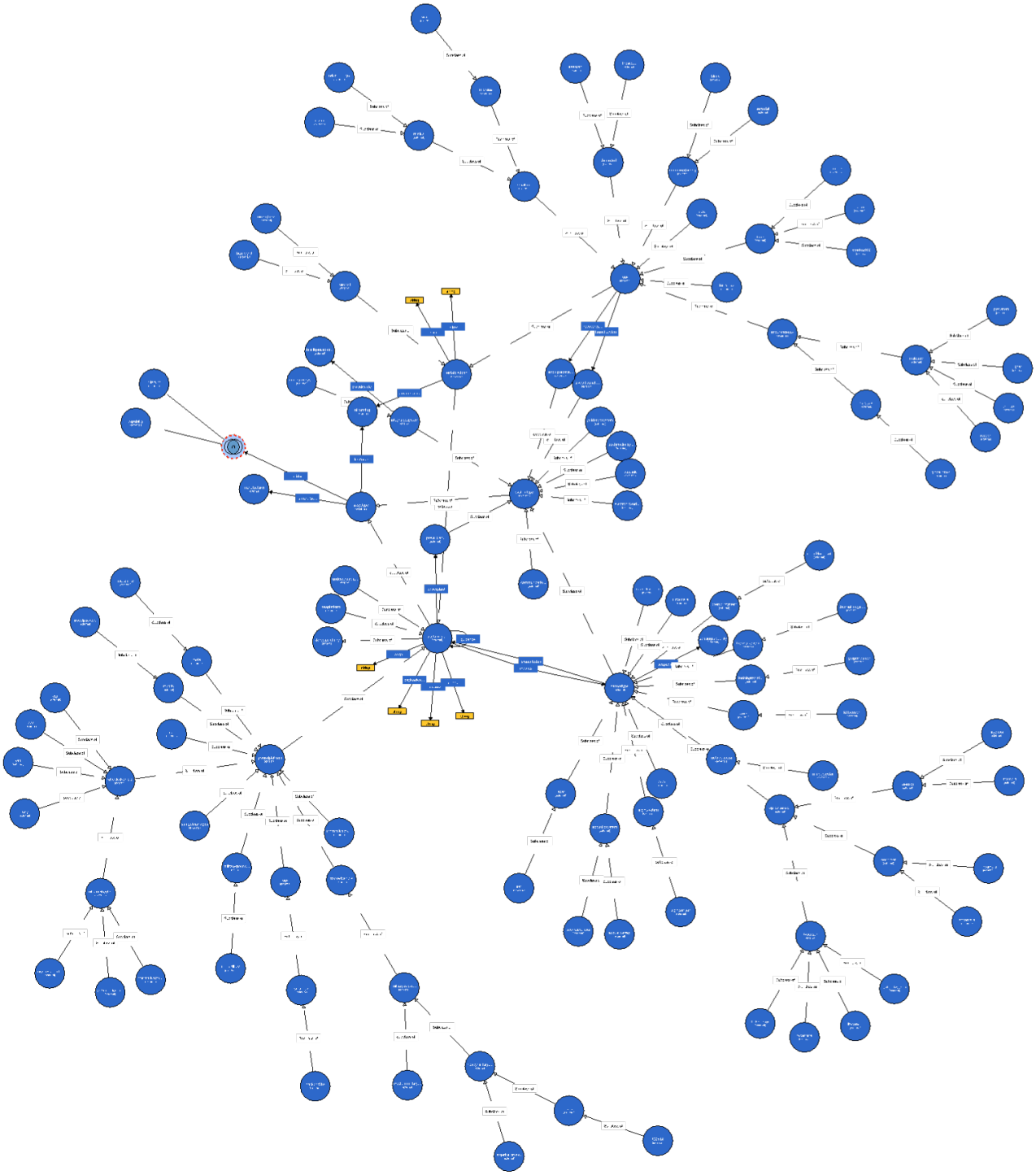

Figure 2. Concepts (classes) converted to OWL from the ISTAR CE model

4. RELATED WORK

Donohue et al[17] use the same ITA Controlled English based conversational mechanism to outline the potential for ontology development but with more of a focus on OWL ontologies, positioning the role of the CNL and conversation as

a simple human-friendly mechanism for gathering ontology extensions from the field rather than being the main semantic representation format. In their work they find good potential for such an approach but are unable to demonstrate that a language such as CE is required to achieve such a solution. Given the necessary focus on the OWL representations and subsequent OWL outputs for their work it is unsurprising that this conclusion is reached, since the role of CE in that context is very much just a simple human-friendly secondary representation of the "master" OWL ontology in which case any readable text could be used without the need for it to be based on a formal semantic representation. In the work described in this paper (and much of our previous research) we position the role of CE as both the human-friendly representation language and the formal machine format. This therefore explains the difference between the two approaches and therefore the difference in importance of the role of a language such as CE within the solution: In Donohue et al the formal representation is OWL and the role of CE is simply to provide a human-friendly representation format with which the user can have conversational interaction with the system, whereas in this paper the formal representation is CE, with the OWL components being generated directly from the CE knowledge base. The distinction is subtle but in the case where CE is the language used for formal knowledge representation as well as the human-friendly information exchange language we believe the potential for building increasingly rich semantics into the possible ontology extensions from the end user in increased significantly.

In addition to our on-going research into the ITA CE language there is a rich field of work into Controlled Natural Languages in the context of the Semantic Web[18]. The purposes of the languages vary, as do the specific implementations, style of language and intended operational usage. In our ITA CE research we have tried to create a language that is different from others in this field, both in terms of complexity, expressivity and intended usage. In this paper we do not attempt to position ITA CE against these other languages however it is likely that the mechanism for capturing ontology extensions from end users in the field could be achievable with many of these other languages, assuming that they are able to be used directly as the formal semantic representation, and that they are able to be embedded into a higher level natural language conversational interaction mechanism.

5. CONCLUSION AND FUTURE WORK

In this paper we have outlined the potential value of a human-machine conversational system that enables non-technical end users in the field to make limited extensions to existing ontologies. Examples have been given that show the potential power of such an approach even with a deliberately limited subset of possible ontological actions. Since this mechanism uses ITA Controlled English (CE) as the basis we also demonstrate a simple technique for converting from CE knowledge bases into standard OWL ontologies and showcase an example implementation by converting an existing complex ISTAR ontology to OWL for consumption by standard OWL tooling such as the WebVOWL ontology visualiser.

In terms of future work: we believe that there is significant potential to be gained in investigating techniques involving the OWL metamodel, but these are not reported here as they have not yet been formally investigated. However, if output is desired in different OWL serialization formats, it may be useful to investigate if the OWL metamodel can be modelled in CE, and rules devised to translate the CE concepts, properties and instances into instances of this representation of the OWL metamodel. This would introduce an abstract representation of the OWL equivalent to the CE present in the system. This representation could then be traversed by a number of agents, each of which could serialize it to a different format (e.g. RDF/XML, N3, Manchester Syntax). Furthermore, simple CE rules (say those that represent reflexive, symmetric and transitive relations) could possibly be translated from simple parsing of the rule text into appropriate OWL elements. In this same area we also concluded that a possible investigation into the translation of CE rules into a rules language such as the Semantic Web Rule Language (SWRL)[16] could be of some value. SWRL is designed to provide a rules language for an OWL knowledge base and the existing CE rule implementation will have a close semantic relationship to the SWRL language.

Acknowledgement

This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. The authors would also like to extend their thanks to Dr Paul Smart from the University of

Southampton for his earlier work in the field of Controlled Natural Languages which continues to guide and inspire the ongoing ITA Controlled English research and development work.

REFERENCES

[1] Kuhn, T. (2014). A survey and classification of controlled natural languages.*Computational Linguistics*, *40*(1), 121-170.

[2] Braines, D., Mott, D., Laws, S., de Mel, G., Pham, T., & Giammanco, C. (2013). Controlled english to facilitate human/machine analytical processing.*Proceedings of SPIE Defense, Secuirty and Sensing*, *8758*, 8758-13.

[3] Preece, A., Braines, D., Pizzocaro, D., & Parizas, C. (2014). Human-machine conversations to support multi-agency missions. *ACM SIGMOBILE Mobile Computing and Communications Review*, *18*(1), 75-84.

[4] Preece, A., Gwilliams, C., Parizas, C., Pizzocaro, D., Bakdash, J. Z., & Braines, D. (2014, May). Conversational sensing. In *SPIE Sensing Technology+ Applications* (pp. 91220I-91220I). International Society for Optics and Photonics.

[5] Gruber, T. R. (1994). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Journal of Human Computer Studies , 43(5/6): 907–928.

[6] McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation*, *10*(10), 2004.

[7] Shadbolt, N., O'Hara, K., Berners-Lee, T., Gibbins, N., Glaser, H., & Hall, W. (2012). Linked open government data: Lessons from data. gov. uk. *IEEE Intelligent Systems*, *27*(3), 16-24.

[8] Preece, A., Pizzocaro, D., Braines, D., Mott, D., de Mel, G., & Pham, T. (2012, July). Integrating hard and soft information sources for D2D using controlled natural language. In *Information Fusion (FUSION), 2012 15th International Conference on* (pp. 1330-1337). IEEE.

[9] Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., & Hart, G. (2008, April). A comparison of three controlled natural languages for OWL 1.1. In *4th OWL experiences and directions workshop (OWLED 2008 DC), Washington* (pp. 1-2).

[10] Preece, A., & Sieck, W. R. (2007). The international technology alliance in network and information sciences. *Intelligent Systems, IEEE*, *22*(5), 18-19.

[11] Mace, J., Mott, D., Braines, D. (2011, May) Supporting OWL-DL in ITA Controlled English. https://www.usukita.org/node/1716

[12] Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., ... & Smith, M. (2009). OWL 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, *27*(65), 159.

[13] Preece, A., Pizzocaro, D., Braines, D., & Mott, D. (2012, May). Tasking and sharing sensing assets using controlled natural language. In *SPIE Defense, Security, and Sensing* (pp. 838905-838905). International Society for Optics and Photonics.

[14] Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating semantic web contents with protege-2000. *IEEE intelligent systems*, (2), 60-71.

[15] Lohmann, S., Link, V., Marbach, E., & Negru, S. (2014). WebVOWL: Web-based visualization of ontologies. In *Knowledge Engineering and Knowledge Management* (pp. 154-158). Springer International Publishing.

[16] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, *21*, 79.

[17] Donohue, B., Kutach, D., Ganger, R., Rudnicki, R., Pham, T., de Mel, G., ... & Smith, B. (2015) Controlled and Uncontrolled English for Ontology Editing. In Semantic Technology for Intelligence, Defense and Security (STIDS)

[18] Smart, P. R. (2008). Controlled natural languages and the semantic web.