Plan ontology and its applications

Lee Chew Hung,   Lee Hian Beng,   Ng Gee Wah,   How Khee Yin
Centre for Decision Support
DSO National Laboratories
20 Science Park Drive
Singapore 118230, Singapore
{lchewhun, lhianben, ngeewah, hkheeyin}@dso.org.sg

Abstract – *An ontology as a formal specification of a shared conceptualisation contains the consensual knowledge of a domain in a machine-readable form. The plan ontology, that we developed, captures the knowledge found in the domain of military planning like organisations, tasks and relations such as task assignment. The design of the ontology was guided by two principles: clarity and modularity. These principles led to human-readable definitions organised as small modules encoded in DAML+OIL, a description logics-based ontology language. The plan ontology was used to structure textual-based plan documents by transforming the information extracted from the documents into ontological models. The models or plan instances were used in a variety of applications like plan navigation and plan retrieval. These applications illustrate the utility of an ontology in plan comprehension and information retrieval from the plan.*

Keywords: Ontology, document retrieval, graph visulization, information extraction.

1 Introduction

According to Gruber, an ontology is the formal specification of a shared conceptualisation [1]. Thus it is an artefact that expresses an intensional model in an explicit and machine-readable form. The ontology contains the consensual knowledge of a domain, capturing the concepts, the relationships between concepts and the axioms used by a group of people working in the domain. In the domain of planning, there are concepts like actor and action, relations like the assignment of an action to an actor and axioms defining for example the starting time of an action must be less than its finishing time.

The strength of an ontology lies in its design and different researchers have proposed different guidelines for good ontology design [2, 3, 4]. Essentially there are two core principles in their proposal: clarity and modularity.

Clarity is the principle of communication. An ontology is basically a communication artefact between its architect and its users (human and machine). As such, the ontology should be well documented and its purpose is clear to its reader. Each ontological element should be named in a descriptive manner that reflects its intended usage. Relations and formal semantics should be stated where possible to provide further clarity to the terms.

Modularity is the principle of management. The ontology should consist of small, internally coherent components. Component ontologies developed in this fashion tend to be self-contained and easier to reuse as they suffer from less ontological commitments than would a large and monolithic ontology.

Developing a domain ontology involves identifying the terms used in the domain from sources like interviews with experts and documents [5]. These terms are divided into concepts and relations and organized into a graph structure where the concepts form the nodes and relations form the arcs. The resultant ontology graph can be validated by techniques like OntoClean [6] that exposes inappropriate and inconsistent modelling choices. The ontology can also be validated using competency questions [7, 8] stated at the start of the development process. Competency questions are formal queries whose answers can be derived from the ontology and its instances.

The rest of the paper is organised as follows. Section 2 provides an overview of the development of the plan ontology and shows fragments of the resultant component ontologies. Section 3 looks at the transformation of unstructured textual plan documents into a set of ontological instances usable in the applications described in Section 4 and Section 5. Section 6 contains a summary of the main points in this paper and provides an overview of the future efforts.

2 Plan Ontology

We applied the Noy & McGuiness methodology [5] augmented with OntoClean analysis and competency questions to the development of the plan ontology with the intention of creating a rich representation that can be used to describe military plans. The result is an ontology containing concepts related to military organisation, military process, time and geographic feature. It also contains the relationships between some of these concepts like `begin`, `end` and `assignedTo`. Fig. 1 and Fig. 2 show part of this ontology.

The concepts in the plan ontology are specialised from the upper ontological concepts found in the Suggested Upper Merged Ontology (SUMO) [9]. An upper ontology contains general ontological entities that are relevant to many domains. For example, the SUMO concept `IntentionalProcess` can be used to describe an economic process like trade as well as military processes like military campaign and military operation. Hence we have the following definitions in the plan ontology written using DAML+OIL:

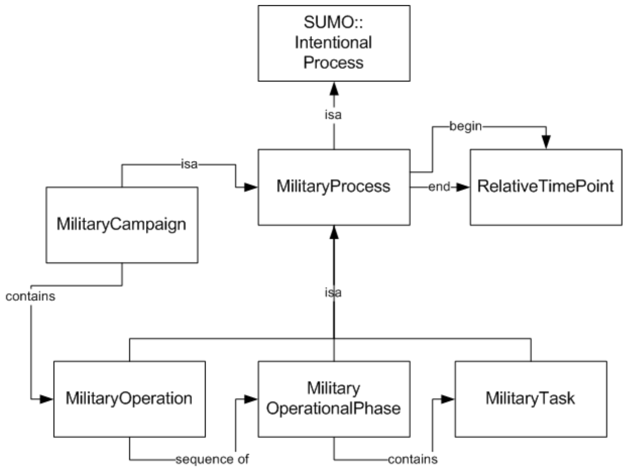


Fig. 1: Plan ontology - MilitaryProcess

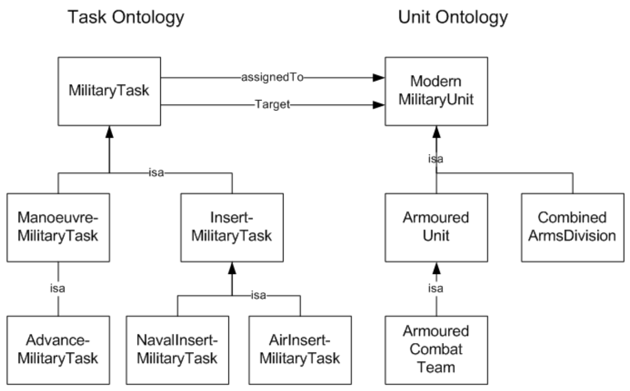


Fig. 2: Plan ontology - MilitaryTask & ModernMilitaryUnit

```
<daml:Class rdf:ID="#MilitaryCampaign">
  <rdfs:subClassOf>
    <daml:Class rdf:about="#MilitaryProcess"
/>
  </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:ID="#MilitaryProcess">
  <rdfs:subClassOf>
    <daml:Class rdf:about="&SUMO;Intentional
Process"/>
  </rdfs:subClassOf>
</daml:Class>
```

DAML+OIL is a Description Logics-based ontology language [10]. It uses XML syntax as a means of serialisation and is developed upon the Resource Description Framework (RDF) and RDF Schema. The W3C Web Ontology Language (OWL) [11], a recent W3C Recommendation released in February 2004, is an improvement to DAML+OIL. It defines clearer semantics and introduces three levels of language (OWL-Lite, OWL-DL and OWL-Full) with each successive level being more expressive and inferencing being less decidable. At a later stage, the plan ontology would be ported to OWL-DL.

Following the principle of modularity, the plan ontology is organised into a set of smaller ontologies like the Task ontology and the Unit ontology with relations being defined among the smaller ontologies (Fig. 2). This allows us to have the flexibility of reusing part of the plan ontology in other applications by importing and extending only the required component ontologies.

3 Structuring plans using ontologies

Military plans are typically prepared based on an assessment of the impending situation before hostilities erupt. These plans are drawn graphically as well as written in a textual form detailing the plan. The prepared plans are then set aside to be retrieved as the situation clarifies.

In a traditional document retrieval system, the textual plans would be passed through an indexing module to extract keywords tagging the documents. However keyword-based retrieval often results in many irrelevant documents being retrieved as it is not sensitive to the semantic structure of the documents. A document is retrieved if it contains the keywords even though the keywords occur in different parts of the document. We suggest a solution using a richer representation like an ontology to capture the semantic structure of the document and increase the precision of the retrieval. We will discuss the retrieval algorithm in section 5.

The process of retrieving plans or documents using ontology is similar to that of traditional document retrieval. First, the plans are passed through an indexing module to reduce the plans into a set of ontological entities. Next these entities are kept in an ontology store. Applications can then make queries to the ontology store and retrieve appropriate ontological entities. These entities can either represent a fragment of a plan or they can point to the source. Fig. 3 illustrates the process of turning plans into ontological entities that can be used by a variety of applications.

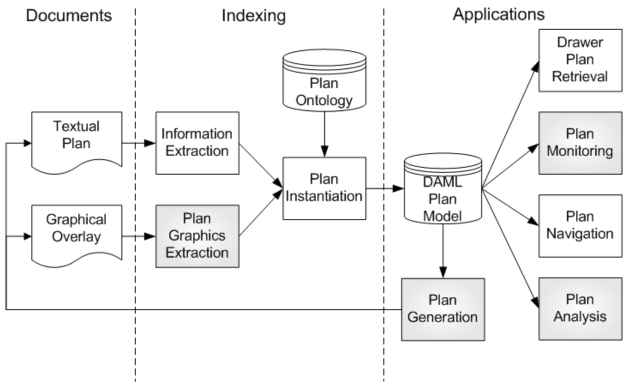


Fig. 3: Process for using plans as ontological entities

Information Extraction (IE) is applied at the start of the indexing process on the textual plans to extract relevant entities. A typical IE workflow using NLP (Natural Language Processing) techniques is shown in Fig. 4. In this workflow, IE starts by segmenting the input text. Each segment is further broken down into sentences and clauses that are in turn broken down into tokens. After the tokens are generated, they are tagged with a part of speech (POS) tag that identifies the token as a noun, a verb, an adjective or other parts of speech.

Next, morphological analysis is applied to each token to extract the root form of the word. The root form makes it easier to handle variations of a word like (advancing, advances and advance) by reducing the various words into a single root form. Collocation analysis is used to group tokens that form a phrasal expression like “take off” into a single token. Noun phrases and verb groups are identified next using a phrase bracket recogniser. The subsequent step in the workflow (Name Entity Recognition) identifies words or phrases belonging to these classes: organisation, person, location, date, time, money and percentage. Finally, a semantic tag (e.g. LOCATION, TIME, AGENT) is assigned to every noun phrase identified as a name entity.

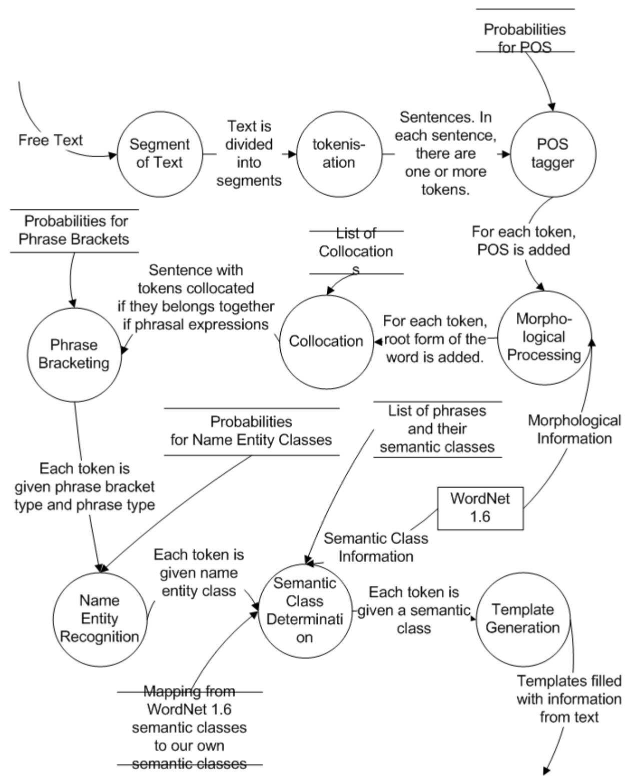

Probabilities for POS
Free Text
Segment of Text
Text is divided into segments
tokenis-ation
Sentences. In each sentence, there are one or more tokens.
POS tagger
Probabilities for Phrase Brackets
List of Collocations
For each token, POS is added
Sentence with tokens collocated if they belongs together if phrasal expressions
Phrase Bracketing
Collocation
For each token, root form of the word is added.
Morpho-logical Processing
Probabilities for Name Entity Classes
List of phrases and their semantic classes
Morphological Information
Each token is given phrase bracket type and phrase type
WordNet 1.6
Semantic Class Information
Each token is given name entity class
Name Entity Recognition
Semantic Class Determination
Each token is given a semantic class
Template Generation
Mapping from WordNet 1.6 semantic classes to our own semantic classes
Templates filled with information from text

Fig. 4: Information Extraction workflow

With all the information gathered from the various steps in the IE workflow, the IE engine then fills the slots in predefined templates. Each template specifies the slots to be emitted and the semantic classes of the value used to fill each slot. The output of the IE engine is a document containing a set of records. Each record created based on the templates contains key–value pairs. The first word on each line is the *key* and the rest of the line is the *value* of the *key*. An example of the record emitted by the IE engine is given in Figure 6.

The extraction of task precedence (temporal dependency) and subtask relationship (logical dependency) is aided by the format of the input documents. Each plan document has a strong outline structure that illustrates the task precedence and subtask relationship (see Figure 5). The first level of the outline describes the operational phases and the second level describes tasks that belong to a particular operational phase. The main tasks are obtained from the first sentence of each paragraph. Subtasks of a main task are described in the rest of the sentences in the paragraph or they can be extracted from keywords like “and” or “with”. For example “Unit X is to advance and secure Alpha” has two subtasks: “advance into Alpha” and “secure Alpha”. The name of a task is derived from keywords describing the task. These keywords are dependent on the operational concepts followed by the planners but certain basic concepts like “attack”, “secure” and “defend” are constant.

- Phase 1
  - Task 1
  - Task 2
- Phase 2

  ⋮

Fig. 5: Plan document structure

```
Action PLAN-A-P1-P1
Annotation moving 1 x Bn (-) to FOOLAND
Location FOOLAND
Name moving
End 1
Begin 0
Actor 1 INF BN
SubAction PLAN-A-P1-P1-P1
Next PLAN-A-P1-P2
```

Fig. 6: Sample IE output

We read each record from the IE output and map the entities described by the record to concepts and relations found in the plan ontology. For example if we see `INF BN` in the slot of a record, we would map it to the concept `#InfantryBattalion` and create an instance of the mapped concept. The *key* of each record is mapped to a relation in the plan ontology. In Figure 6, the `#InfantryBattalion` is `#MilitaryTask.assignedTo` the `#MilitaryTask` represented by the record. As the record references other records (e.g. actions and subactions) whose types are unknown at the point of processing, typeless instances are created first. The type of these instances is revised when sufficient information is available to determine their types. Jena [12] is used to hold and output the instances into a DAML+OIL file with each DAML+OIL file representing an instance of a military plan.

4 Plan Visualisation

The goal of plan visualisation is to present the plan in an intuitive manner that facilitates understanding and analysis. A popular form of visualisation is the Gantt chart where the resources and activities are laid out on a time-line. This allows the user to track the resource utilisation (e.g. is the unit overextended in an operation) as well as the dependencies between activities (e.g. how does a delay in one operation impact the timing of another operation). Another popular presentation form used by military planner is that of the graphical overlays containing units and manoeuvre arrows. This allows the user to grasp the bigger picture of the military operation through the graphics depicting the various axes[1] of advances and the units involved in the operation.

The ontological structure of the plan produced in section 3 allows us to build a different visualiser that is focused on the relationship between entities. This visualiser loads the plan using Jena and uses the Jena's API to retrieve the ontological entities that are added to a graph toolkit for display. The graph toolkit that we used is the TouchGraph GraphLayout written by Alexander Shapiro (http://www.touchgraph.com). The ontology visualiser allows us to view different aspects of a plan (e.g. the geographical features or the units involved) and study the the relationships or dependencies between different entities. The visualiser also allows the user to drill down and expose the relationships held by an entity. For example, we can see the number of tasks assigned to a particular unit.

A search dialog in the visualiser allows the user to locate nodes using part of the URI of the class or instance. By using the `Find Next` and `Find Prev` button, the user can also cycles through the matches and the visualiser will display the matched node in the centre of the view.

Fig. 7 is an illustration of how the ontological structure facilitates understanding of the plan. It shows the plan ECA-A contains five demolition preparation tasks. The URIs of the five tasks are also informative. In this figure, the first word after the word 'ECA-A-' is the phase number. We can see that phase 2 has two demolition preparation tasks and phase 3 has three demolition preparation tasks. Expansion of these nodes may reveal other information (e.g. is the same location being targeted for demolition by more than one unit).

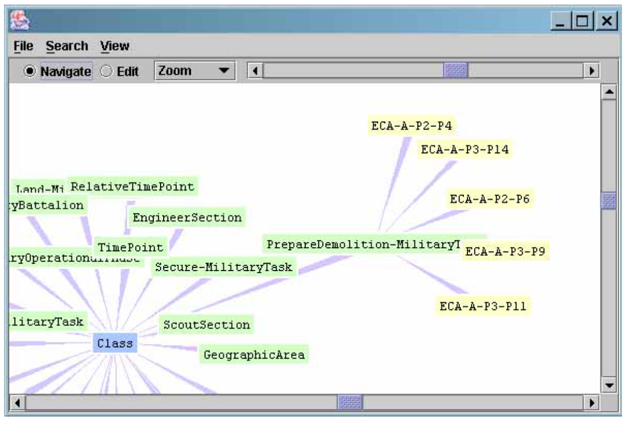
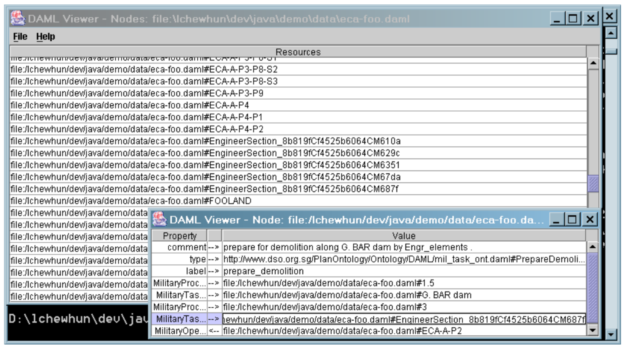


Fig. 7: Screen capture of Graph-based Viewer

Colours are also used to differentiate the various ontological entities. Classes are displayed in green, instances in yellow and relations in purple. The colours are customisable through a configuration file in XML so that different classes and relations have unique colours. For example, we can customise the display such that a "target" relation is highlighted in red.


DAML Viewer - Nodes: file:\lchewhun\dev\java\demo\data\eca-foo.daml

```
Resources
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P3-P8-S2
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P3-P8-S3
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P3-P9
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P4
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P4-P1
file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P4-P2
file:/lchewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM610a
file:/lchewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM629c
file:/lchewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM6351
file:/lchewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM67da
file:/lchewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM687f
file:/lchewhun/dev/java/demo/data/eca-foo.daml#FOOLAND
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
file:/lchewhun/dev/java/demo/data
D:\lchewhun\dev\jav
```

DAML Viewer - Node: file:/lchewhun/dev/java/demo/data/eca-foo.da...

Property		Value
comment	-->	prepare for demolition along G. BAR dam by Engr_elements .
type	-->	http://www.dso.org.sg/PlanOntology/Ontology/DAML/mil_task_ont.daml#PrepareDemoli...
label	-->	prepare_demolition
MilitaryProc...	-->	file:/lchewhun/dev/java/demo/data/eca-foo.daml#1.5
MilitaryTas...	-->	file:/lchewhun/dev/java/demo/data/eca-foo.daml#G. BAR dam
MilitaryProc...	-->	file:/lchewhun/dev/java/demo/data/eca-foo.daml#3
MilitaryTas...	-->	ewhun/dev/java/demo/data/eca-foo.daml#EngineerSection_8b819fCf4525b6064CM687f
MilitaryOpe...	<--	file:/lchewhun/dev/java/demo/data/eca-foo.daml#ECA-A-P2

Fig. 8: Screen capture of text-based Viewer

A related application is the generic DAML Viewer available from the DAML programme's website (http://www.daml.org). The DAML Viewer is a text-based browser of DAML+OIL files. It presents the information in an ontology file as a series of frames (or a table of attribute-value pairs). The viewer allows the user to drill down or navigate through the links formed by the relationships between resources. This viewer is especially suitable for large ontologies as it presents a limited set of information from the ontology in each frame instead of all the information.

Figure 8 shows the plan ECA-A displayed in the DAML Viewer. DAML Viewer starts by listing the resources (list of URIs) available in the DAML ontology file. As the user clicks on each resource, a new window (resource information window) is displayed about the resource and the relationships it holds with other resources. The resource information window lists each relation or property, the direction of the relationship and the resource that is related to the resource being displayed in this window. In the figure, a task corresponding to the text "prepare for demolition along G. BAR dam by Engr_elements." is displayed. From the display we can see that this task is related to #ECA-A-P2, the type of the task (#PrepareDemolition-MilitaryTask), the location of this task (#G.BAR dam) and other information.

4.1 Issues

An issue confronting ontological systems is handling large ontologies. An initial implementation of the visualiser displayed all the ontological entities in the plan resulting in a mess of nodes and links. Moreover due to the dynamic nature of the graph toolkit that attempted to balance the graph display causes the display to be caught in a flux resulting in serious system performance.

The current approach adopts an incremental display that starts by displaying a graph of the DAML+OIL classes present in the plan. As mentioned previously, functionalities are provided for expanding and collapsing graph nodes.

[1]An axis is a line of communications. In the context of unit movement, an axis is a path permitting movement like a road or dirt track

As the user expands each node, concepts or instances with relationships to the selected node are added to the graph display. In addition, if the additional nodes have relationship to existing nodes in the display, links are drawn between them achieving an ontological web. Collapsing nodes is the mechanism developed to allow the user to manage the display of a large ontology. Together with the search functionality, the visualiser allows the user to localise and focus on part of the ontology relevant to his interest, hiding the irrelevant parts.

5 Plan Retrieval

In case-based reasoning (CBR), a reasoner uses past experiences to help in formulating a solution to the current problem. The reasoner first performs a search through his case database (or case base for short) and retrieves cases with similar features to the current situation. The solutions applicable to these retrieved cases are considered next. The reasoner may adapt or revise the old solutions to fit the current problem or he may use the old approaches to critique a new solution. The key idea behind CBR is that similar problems occur regularly and have similar solutions [13].

At the heart of any CBR system is a library of stored cases. When a new problem arises, the system matches the features of the new problem against the cases stored in the library and retrieves the cases with good matches. The solutions for the retrieved cases are then adapted to the new problem on hand and the new solution with the new problem are stored into the library for future references.

This is similar to the process military planners engage in while preparing drawer plans. Drawer plans are prepared based on an initial assessment of the general situation (typically before hostility breaks out) and these plans are kept aside for future revision. As the situation develops, the planner retrieves the plan that has the closest fit to the current situation and adapts it to fit the current situation. Automation of this process resulted in a drawer plan retrieval application.

In this application, each plan is stored in a plan base together with a set of features representing the background of the plan, the disposition of the military forces and salient geographic features. The stored plans and their features are then matched against user-supplied search criterions. The stored plans are ranked using a similarity score computed from the match statistics and presented to the planner for further actions.

5.1 Matching Algorithm

For a given set of criterions, the application will match the concepts (Algorithm 1), instances (Algorithm 2), and relations (Algorithm 3) found in the criterions. The similarity score (normalised to a scale from 0 to 1) used to rank the plans in the plan base is computed as an average of the scores obtained from the three algorithms. A score of 1 indicates a perfect of a set of features against the user criterions and a score of 0 indicates that no matches are obtained. A similarity score that tends toward the value of 1 indicates a greater degree of match and a score that tends toward the value of 0 indicates a lower degree of match.

```
input   : criteria, features
Data    : matches

foreach DAML class in the criteria do
    if it exists in the features or it is the type of an instance in the features then
        increment matches

return matches and number of DAML classes in the criteria
```

Algorithm 1: Compute concept matches

```
input   : criteria, features
Data    : matches

foreach DAML instance in the criteria do
    if it exists in the features then
        increment matches

return matches and number of DAML instances in the criteria
```

Algorithm 2: Compute instance matches

```
input   : criteria, features
Data    : exact_matches, partial_matches

foreach DAML property p that maps a set of classes (domain D) to a set of classes (range R) in the criteria do
    if p exists in the features with a larger domain $D'$ ($D \subset D'$) and a larger range $R'$ ($R \subset R'$) then
        increment exact_matches

    else if there exists in the features an instance from D and an instance from R that are related by p then
        increment exact_matches

    else if p exists in the features with a larger domain $D'$ or a larger range $R'$ then
        increment partial_matches

    else if there exists in the features an instance from D or an instance from R related by p to some other instances then
        increment partial_matches

return exact_matches, partial_matches and number of DAML properties in the criteria
```

Algorithm 3: Compute relation matches

The implemented algorithms provide us with a more precise retrieval compared to simple keyword retrieval. Consider a plan A that describes an operation involving an infantry battalion defending hill_193 against an attacking armoured combat team. If our query is looking for an "infantry battalion attacking hill_193", then keyword retrieval would score plan A as 100% since all three keywords "infantry battalion", "attacking" and "hill_193" are mentioned in the plan. On the other hand, the plan retrieval algorithm would not score this plan as 100% as the subject of the attacking task in the plan is an armoured combat team and not the infantry battalion as specified in the query.

An enhancement to the application is to allow successive refinement of the results. After retrieving all the plans involving an infantry battalion defending hill_193 against an attacking armoured combat team, the user may also like to narrow his search by specifying that a reserve force of an armoured infantry company is proceeding to relieve the defending infantry battalion. This would involve running the matching algorithms over the initial result set with the new criterion.

5.2 Pattern-based Retrieval

We also provided pattern-based retrieval using Algorithm 4. Given a pattern of the form $< subject, relation, object >$, the application extracts entities with the relation specified in the pattern from each plan. We can also specify a wildcard in any or all of the positions in the pattern that will bind to any entity. The key idea is to expand each part of the pattern to either a concept or an instance and output the resultant triple if the plan contains it.

```
input   : pattern $< subject, relation, object >$, features
if the relation is a wildcard then
    expand the wildcard with the set of relations found in the features
foreach new pattern p $< subject, r, object >$ obtained after the expansion of the relation do
    if subject is a wildcard then
        expand the wildcard with all the subjects found in the features that are related by r to an entity
    if object is a wildcard then
        expand the wildcard with all the objects found in the features that are related by r to the subject
    if the subject or the object is a concept then
        add new patterns with the concept replaced by instances of the concept found in the features
Verify and remove each pattern that is not found in the features.
return list of bindings for the input pattern
```

Algorithm 4: Match ontology pattern

6 Conclusion and Future Works

This paper covers the use of a plan ontology to structure textual plans. We have presented two applications that can exploit the ontological structure for better presentation and retrieval of information. The first application used the ontological structure to highlight the relationships between different plan entities. The second application used the plan ontology to structure the queries and provide a more precise retrieval of stored plans as compared to the approach using keywords queries.

Other possible works include plan monitoring, plan analysis and plan generation. Plan monitoring compares the situation picture extracted from a real-time situation database and checks the status of a plan, whether it is in progress or if there is a delay in the plan. Essentially, the plan recognition algorithm in the plan monitoring application matches the plan fragment represented by the situation picture against the collection of plans.

Plan analysis is a broad area that includes activities like monitoring resource allocation and computing space-time dependencies. Since the plan ontology contains the concepts of unit, task and time and their relationships, the algorithm turning the instances found in the DAML plan model into a Gantt chart would look out for tasks and their subtasks as well as the starting time of a task and the ending time of a task, arranging the tasks according to the "less" relations defined among the time instances. The ontological entities in the plan model also make it possible to define analysis algorithms that look for specific patterns in the model. For example, we may look for units that have been assigned more than one task in a single phase. We may also look for a road intersection that is the location of many manoeuvre tasks as this may point to a gridlock situation.

Plan generation looks at the transformation of the plan model into a visual presentation like a graphical overlay. If we can also extract plan entities and instantiate plan ontology entities from the graphical overlay, then we can look at the re-planning process. In the re-planning process, the user would make an initial plan by drafting a graphical overlay using the graphical tools from a C2 application. The user's interaction with the system can be captured into a plan model that is instantiated from the concepts and relations in the plan ontology. Other sources or users may have inputs to the plan that changes it. This updates are made to the plan model and it is then used to generate the graphical overlay for the user's consideration. The user can then alter the plan through the C2 graphical interface thus completing the replanning loop.

Acknowledgements

We would like to thank the Defence Science and Technology Agency of Singapore for their funding of this research. We would also like to thank the anonymous reviewers for their invaluable comments and hope that the revised paper has addressed their concerns. Last but not least, we are grateful to Dr. Rodney Teo from DSO National Laboratories for his critical review of the revised paper.

References

[1] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, 1993.

[2] Rudi Studer, Richard V. Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[3] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[4] G. van Heijst, A. Th. Schreiber, and B. J. Weilinga. Using explicit ontologies in kbs development. *International Journal of Human and Computer Studies*, 46(2-3):183–292, 1996.

[5] Natasha F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Stanford technical report, Stanford University, 2001.

[6] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with ONTOCLEAN. *Communications of the ACM*, 45(2):61–65, February 2002.

[7] Michael Gruninger and Mark S. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, Trondheim, Norway, 1994.

[8] Michael Gruninger and Mark S. Fox. Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montrreal, 1995.

[9] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Maine, USA, 2001.

[10] Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. DAML+OIL: An ontology for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, September/October 2002.

[11] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. OWL web ontology language guide. W3C Recommendation, W3C, 2004. http://www.w3c.org/TR/owl-guide/.

[12] Brian McBride. Jena: Implementing the RDF model and syntax specification. In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb 2001*, Hong Kong, China, 2001.

[13] David B. Leake. CBR in context: The present and future. In David B. Leake, editor, *Case-Based Reasoning - Experiences, Lessons, & Future Directions*, chapter 1. AAAI Press, 1996.