ENGINEERING TIME IN AN ONTOLOGY FOR POWER SYSTEMS THROUGH THE ASSEMBLING OF MODULAR ONTOLOGIES

Jorge Santos, Luís Braga
Departamento de Engenharia Informática, Instituto Superior de Engenharia, Porto, Portugal

Anthony G. Cohn
School of Computing, Leeds University, Leeds, U.K.

Keywords: Power Systems, Ontologies, Knowledge Engineering, Temporal/Spatial Reasoning and Representation.

Abstract: In this paper we investigate how timeless ontologies such as DFault, an ontology for fault diagnosis in power transmission networks can be re-engineered to include temporal entities. We propose a methodology, FONTE (Factorising ONTology Engineering complexity), that allows this complex process to be factored by dividing the problem into parts: modelling the domain concepts ontology (atemporal and aspatial), modelling or acquiring the temporal and /or spatial ontology, and finally producing the target ontology by assembling these modular ontologies via a semi-automated process.

1 INTRODUCTION

In recent years many technologies have been developed allowing the capture and transmission (*e.g.*, smart sensors networks and wireless communications) of information about the state of a wide range of devices connected to networks scattered over large geographical areas (*e.g.*, power transmission grids). The emergence of technology standards lowered the prices for these devices/technologies and eased its integration with existing systems. The result is that the data owners are overloaded with large quantities of heterogeneous information. One major challenge therefore consists of developing applications able to provide end-users with a system overview in a human readable, relevant and consistent manner, so as to interpret and manage the system being monitored. There are at least two issues obstructing this goal: the information heterogeneity and the requirement for appropriate models for spatial and temporal knowledge (STK). The modelling of STK in intelligent systems is a complex process. This work proposes a semi-automatic method, FONTE (Factorising ONTology Engineering complexity), based on the assembling of multiple ontologies in order to obtain the target ontology. This method tackles both the above problematic issues: information heterogeneity and modelling STK. The first is addressed through the use of ontologies as building blocks, given that ontologies are shared and common agreed models about a specific domain. The second issue is addressed through the use of a set of rules that drives the process of assembling orthogonal categories, like space and time, in a semi-automatic way, releasing the knowledge engineer and the experts from the rendering of intricate concepts related to complex theories of time and space.

2 FONTE METHODOLOGY

In order to illustrate the assembly process two ontologies will be used as building blocks for the target ontology, a temporal ontology and the timeless domain ontology DFault (www.dei.isep.ipp.pt/∼jsantos/F2C/DFault) for fault diagnosis in power transmission networks.

Modular Ontologies. The assembly process can be used either for the development of ontologies with time from scratch, or for re-engineering existing atemporal ontologies in order to include time. For our case study we have used the time-less DFault ontology that captures the main concepts related to the characterisation of the Portuguese National Electricity Transmission Grid (RNT) (www.ren.pt), and concepts related to fault diagnosis in power transmission networks (previously used to develop the SPARSE system (Vale et al., 2002)).

The DFault ontology (OWL version)includes 92 classes, 30 properties and 30 restrictions. The hierarchy is divided in four sub hierarchies, with root concepts: Event, Entity, Local and Object. The Event class includes the different type of events that may occur in the electric network, such as breaker tripping or power lines going out of service. The Entity class includes the organisations (*e.g.*, suppliers, clients) and persons involved in the management/exploration of the network. The Object class is divided into physical objects, which include the electric network devices and facilities, and abstract objects, which include the faults diagnosis and messages acquired from the SCADA (Supervisory Control And Data Acquisition) system. The Local class represents a hierarchy of physical locations, which are used to spatially characterise the network devices and facilities.

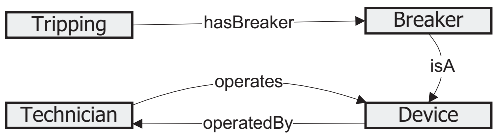


Figure 1: Excerpt of the DFault Ontology.

Fig. 1 presents an excerpt of the DFault ontology that was used in order to elucidate the FONTE method and the Protg plug-in that supports it. A circuit breaker is an electrical device designed to protect a circuit from damage caused by an electrical event such as overload or short circuit. Its basic function is to detect a fault condition and, by interrupting continuity (tripping), to immediately discontinue electrical flow. A circuit breaker can be reset (either manually or automatically) to resume normal operation.

The temporal ontology used in our case study (see Fig. 2 for the UML-like depiction of an excerpt) embodies many concepts such as Instant or Period found in 'standard' ontologies such as OWL-Time (www.w3.org/TR/owltime/) or SUMO (www.ontologyportal.org/) and assumes a standard interpretation, mapping time points and intervals to real numbers and intervals on the real line respectively. A temporal representation requires the characterisation of time itself and temporal incidence, which are represented in our temporal ontology by TemporalEntity and Eventuality, respectively. A further notion, TimedThing, which is used during the assembly process, bridges between temporal concepts and domain concepts .

Temporal Entities. In the temporal ontology we used for the case study there are two subclasses of TemporalEntity: Instant and Period. The relations *before*, *after* and *equality* can hold between Instants,

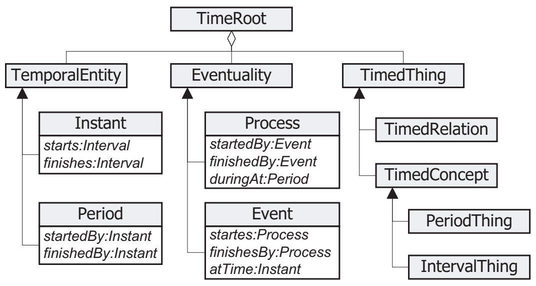


Figure 2: Excerpt of the Temporal Ontology.

respectively represented by the symbols:$\prec$, $\succ$, $=$, allowing an algebra based on points to be defined(Vilain et al., 1989). It is assumed that the *before* and *after* are strict linear, i.e. irreflexive, asymmetric, transitive and linear. The thirteen binary relations proposed in the Allen's interval algebra (Allen, 1983) can be defined easily from $\prec$, $\succ$, and $=$. The *starts* and *finishes* are relations from TemporalEntity to Instant. Also, there are no null duration periods and each period is unique.

Processes and Events. There are two subclasses of Eventuality, Process and Event, in order to be possible to express continuous and instantaneous eventualities, respectively. Event has a relation *atTime* to Instant while Process has a relation *duringAt* to Period. The relations *starts* and *finishes* can be used to state what can start or finish a process.

The Assembly Method. FONTE (Santos and Staab, 2003) allows the targeted complex ontology to be built by factorising concepts into their temporal, spatial and domain (atemporal and aspatial) aspects, and then assembling the temporally/spatially situated entity from these primitive concepts. This is more similar to a Cartesian Product than a union of ontologies. Each of these component ontologies can be built/acquired independently, allowing a factorisation of complexity. The ontologies assembly is performed through an iterative and interactive process that combines two types of inputs: *i)* human assembly actions between the component ontologies; and *ii)* automatic assembly proposals obtained from semantic and structural analysis of the ontologies. This process is propelled by a set of rules and a set of constraints. The set of rules drives a semi-automatic process proposing assembly actions; the set of constraints allows to assess which of the generated proposals are valid.

The proposed methodology divides the task of building a temporalised ontology for power network control and monitoring into the task of constructing simpler sub-ontologies which can be built independently, factorising the problem complexity. The domain concepts ontology DFault (DO) and the temporal ontology (TO) (both described in the previous section) will be used to illustrate the assembly process.

The methodology hence proposes to obtain the target ontology (power network control and monitoring with a temporal) through a semi-automatic process of assembling. The concept DO.Device.Breaker will be linked with TO.TemporalEntity, while the property *DO.Device.Breaker.triggering* will be linked with TO.Process. In this way the instances of DO.Device.Breaker concept will have a time span of use. Instances of *DO.Device.Breaker.triggering* process have a time span in which they occur and additionally the user will be asked to assemble the events that define the beginning and ending of this process (*e.g.*, *DO.Device.Breaker.opening* and *DO.Device.Breaker.closing*). After this step, related concepts, properties and axioms are proposed for assembling, through a cascade process: e.g. sibling concepts of DO.Device.Breaker such as DO.Device.Line or DO.Device.Transformer will be proposed for the assembly process; properties and axioms related to *DO.Device.Breaker.triggering* like *....triggering.monophasic* or *....triggering.triphasic* would be proposed for assembly depending on their characteristic event sequence.

Protg Plug-in. In order to support the iterative and interactive process used in FONTE, a Protg plug-in was developed. An assembly task consists of the definition of the actions to be performed in the target ontology after the performance of an assembly action (*e.g.*, creation of a relation *isA* between a domain and a temporal concept). Due to the characteristics of the platform (Protg), two types of tasks were defined:

Internal Tasks, which allow basic operations to manipulate the ontologies to be performed (*e.g.*, create, delete and modify classes or properties), and provide access to the API functionalities of the Protg platform in a transparent mode;

External Tasks. (also called assembly rules), which are procedures written in a pseudo-code language that includes common program language instructions (*e.g.*, if, then, else) and special keywords (*e.g.*, do, propose, check) whose semantics has been previously provided. In order to facilitate the edition/creation of these tasks, a specific tool supported in a graphic interface was developed (see below).

The FONTE plug-in architecture (see Fig. 3) has different abstraction levels which present several advantages for the knowledge engineer: *i)* the knowledge engineer does not need to know the specifics of the Protg API to manipulate the ontologies. In addition, the Internal Tasks provide an abstraction level between External Tasks and Protg API assuring independency between the External Tasks and the Protg API version; *ii)* the External Tasks may be created/edited during the execution time and do not require the alteration of the application and consequent compilation; *iii)* Different rules set (stored in distinct files) allow different temporal/spatial theories in the assembly process to be used in a flexible way.

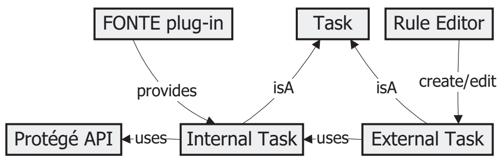


Figure 3: FONTE plug-in architecture.

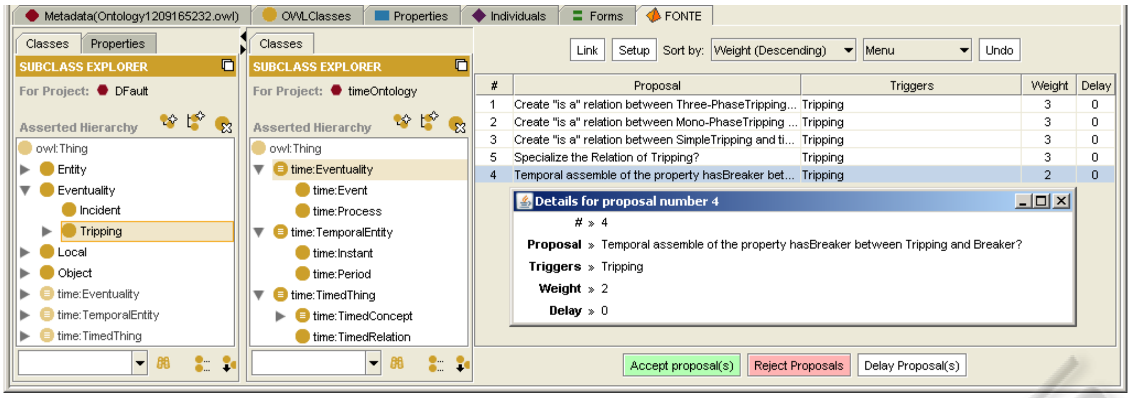
The plug-in provides a set of functionalities, such as: linking concepts of the domain and temporal/spatial ontology; accepting, rejecting or even delaying the execution of a task; and visualising statistics of the assembly process. As depicted in Fig. 4, the plug-in has two panels for the manipulation of ontologies (on the left-hand side) and a list of proposals (on the right-hand side). The panel further to the left contains the domain ontology (DFault, which is timeless and spaceless); from this panel it is possible to access the classes and properties hierarchies. The other panel contains the temporal/spatial ontologies to be used as construction blocks for the production of the target ontology. The list of proposals contains the records of the task instances generated by the system. Details of this list are presented below.

To promote the assembly process the knowledge engineer needs to select the ontologies that will participate in the assembly process as well as the files containing the assembly rules for each ontology; these can be selected using the setup window (triggered by the setup button shown in Fig. 4).

All the tasks that are successfully performed (either triggered manually by user-driven action or automatically by the structural analysis module) are added to a list containing the instance tasks historic. Associated with each task instance proposal there is: a trigger list (the elements that triggered the proposal); the task weight (an indication of the importance of each proposal, which influences the likelhood of proposal acceptance during the assembly process); and a question in natural language (a phrase that summarises the proposal objective, instantiated with the elements contained in the instance task).

As the assembly process progresses, more proposals are generated. If different concepts happen to propose the same task instance, all the elements that have triggered that proposal are included in the trigger list and the proposal weight is increased.

All the proposed task instances are stored in the list of proposals, which can be sorted by different criteria (*e.g.*, id, trigger or weight). The user can then accept, reject, or even delay for later analysis,



#	Proposal	Triggers	Weight	Delay
1	Create "is a" relation between Three-PhaseTripping...	Tripping	3	0
2	Create "is a" relation between Mono-PhaseTripping ...	Tripping	3	0
3	Create "is a" relation between SimpleTripping and ti...	Tripping	3	0
5	Specialize the Relation of Tripping?	Tripping	3	0
4	Temporal assemble of the property hasBreaker bet...	Tripping	2	0

Details for proposal number 4

» 4
Proposal » Temporal assemble of the property hasBreaker between Tripping and Breaker?
Triggers » Tripping
Weight » 2
Delay » 0

Figure 4: FONTE plug-in for Protg.

each of the proposals. In order to avoid overloading the knowledge engineer with useless proposals, rejected proposals are never automatically proposed again(though they may be manually retrieved).

In addition to the functionalities described above, the plug-in also provides statistics about the assembly process (results of the tool performance, including the initial and current status of the domain ontology, the number of tasks that has been initiated by the user and how many proposals have been accepted or rejected). A facility for saving a sequence of performed tasks as a *script file* is also provided to allow a knowledge engineer to easily totally or partially repeat a certain set of tasks.

An application was developed to facilitate the creation of external files of rules. This supports the knowledge engineer through a simple and interactive graphic interface. The file management system provides a graphic visualisation of the rules included in each file and offers various functionalities, such as: sorting the list through different criteria; modifying the order in which the rules are interpreted during the assembly process; visualising the rules in XML or pseudo-code; removing, editing or creating new assembly rules. The knowledge engineer is alerted about potential consistency errors (*e.g.*, non declared variables) or warnings (*e.g.*, to declaring a variable that is not used).

3 CONCLUSIONS AND FUTURE WORK

In this paper we discussed the engineering of time in DFault, an ontology for fault diagnosis in power transmission networks. We proposed FONTE, a method that supports the engineering of complex ontologies including temporal and/or spatial knowledge that allows process complexity to be factored by dividing the problem in parts: modelling the domain concepts ontology (atemporal and aspatial), modelling or acquiring the temporal and/or spatial ontology, and finally producing the target ontology by assembling these modular ontologies through a semi-automated rule based approach.

A Protg plug-in developed to support method FONTE, which allows FONTE to be used in an integrated form in the development of ontologies, was also described. The FONTE methodology works independently of the temporal/spatial theory since it allows different sets of assembly rules to be used for each specific theory. A tool to support the creation/edition of these rule sets was also summarised.

Future work includes: *i)* evaluating the generic characteristics of the proposed method with different spatial/temporal ontologies (including 4D ontologies); *ii)* developing a functionality to predict the impact of the acceptance of a particular proposal; *iii)* improving the generation of automatic proposals during the assembly process; (ii) and (iii) may be achieved through the use of semantic analysis, previously successfully used in diverse processes of ontology engineering (*e.g.*, merging, mapping and alignment).

REFERENCES

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communication ACM*, 26(11):832–843.

Santos, J. and Staab, S. (2003). FONTE - Factorizing ONTology Engineering complexity. In *The 2nd Int. Conf. on Knowledge Capture (K-Cap'03)*, pages 146–153.

Vale, Z., Ramos, C., Faria, L., Malheiro, N., Marques, A., and Rosado, C. (2002). Real-time inference for knowledge-based applications in power system control centers. *Journal on Systems Analysis Modelling Simulation (SAMS), Taylor&Francis*, 42:961–973.

Vilain, M., Kautz, H., and Beek, P. (1989). Constraint propagation algorithms: a revised report. *Readings in Qualitative Reasoning about Physical Systems*.