ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPL05

APPLYING ONTOLOGICAL APPROACH TO STORING CONFIGURATION DATA

M. A. Ilina [1] , P. B.Cheblakov, BINP SB RAS, Novosibirsk, Russia
[1]also at Novosibirsk State University, Novosibirsk, Russia

Abstract

Control systems of large experimental facilities need a great number of heterogeneous interconnected parameters to control software applications. As configuration information grows in volume, it becomes harder to be maintained manually and poses a potential threat to data integrity. To tackle this problem, we applied ontological approach to storing configuration data. Ontology is a formal representation of concepts and relations of the domain of discourse, enriched by rules for inferring assumed knowledge. We designed the ontology that describes the controlling electronics for the double-direction bipolar transfer line K-500, which transports beam from the Injection Complex to colliders VEPP-4 and VEPP-2000 at BINP, Novosibirsk, Russia. We populated the ontology by importing data from existing configuration files of the control system and developed the interface for querying configuration data. The designed storage has several benefits over the conventional approaches. It maintains heterogeneous objects with non-trivial dependencies in centralized form, performs data verification and can be expanded to the diverse ontology describing all information about the facility.

INTRODUCTION

Any large experimental facility uses a huge number of hardware devices and software applications to perform control and maintenance functions. Configuration data that describe this equipment has a convoluted structure, including various devices and their templates, physical and logical connections between them, accompanying cable and inventory information, etc. It is crucial for both physicists and facility operators to have access to these varied parameters; consequently, we should store them in a formalized and structured way and provide user interfaces for data access and modification.

Designing an entity-relation diagram and building a relational database based on it, seems to be a straightforward solution to the problem of storing configuration data. However, relational data model lacks flexibility when it describes complex relations. Apart from having a complicated structure, configuration parameters usually have non-trivial dependencies and interconnections. Reconfiguration of any single facility element calls for a sequence of changes related to other elements and they need to be applied in correct order. These peculiarities make it hard to apply any changes to existing system and avoid data inconsistencies when the control system grows in size and becomes more complex.

Therefore, we turned to more flexible graph data model as a basis of desired data storage. It is capable of reflecting complex relationships between heterogeneous objects, which makes it more effective in expressing interconnected configuration data parameters.

REQUIREMENTS ON THE CONFIGURATION DATA STORAGE

Facility

As a model system, we examined double-direction bipolar transfer line K-500 [1] at Budker Institute of Nuclear Physics (BINP), Novosibirsk, Russia. It connects Injection Complex and collider VEPP-4 by transporting electron-positron beam with a frequency of 1 Hz. 24 electronic devices of 8 types are used for controlling the facility power supplies and measuring their parameters. The electronics is controlled by the modular control system CXv4 [2], which configurations are stored in local files and include information about devices, device templates, control channels and their characteristics.

Offered Solution

To automatize the processes of configuration, documentation and maintenance of K-500 facility control system, we required a centralized data storage, flexible enough to describe diverse collection of facility elements and their relationships and reliable enough to provide multiple user access to data.

We defined the following main problems, which the developed storage should be able to solve:

1. Storing diverse facility data in a centralized way;
2. Avoiding data duplications and inconsistencies;
3. Automating configuration of software applications for control system;
4. Documenting control system information in various forms and views;
5. Tracking changes made by system users.

The desired storage should display a single source of actual information about the control system structure and should be interoperable with other control system software. We therefore needed to develop programming and user interfaces for data access, modification and analysis of stored data.

To achieve these goals, we decided to develop a knowledge base for the facility domain, rather than use a traditional database as configuration data storage and take advantage of semantic analysis possibilities. [3]

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPL05

SEMANTIC MODELLING APPROACH

Knowledge Conceptualization

The main aim that we want to reach when creating an information model, or knowledge base, is *conceptualization* [4] of information, which means expressing it in the form, understandable by computers. This information includes all concepts and objects existing in the domain, possible or actual relations between them, existing data types and concrete attribute values, rules for inferring assumed knowledge and restrictions.

As a means of formalisation, we use *ontologies* [5] – formal representations of knowledge about given domain of discourse. Ontologies represent sets of triplet statements about domain expressed in *OWL* (Ontology Web Language) [6] – computational logic-based language for describing ontologies, built upon RDF (Resource Description Framework). Figure 1 shows examples of statements from K-500 Control System Ontology, which are grouped in categories.

```
# metadata (concepts):
    :CSDevType    rdf:type   owl:Class .
    :hasChannel   rdf:type   owl:ObjectProperty .

# data (individuals and their attributes):
    :ceac124       rdf:type   :CSDevType .
    :ceac124/adc0  rdf:type   :CSDevChannel .

    :ceac124/adc0  :hasDirectionType "input" ;
                   :hasDataType      "integer" .

# relations:
    :ceac124  :hasChannel   :ceac124/adc0 .

# restrictions:
    :belongsToDevType  rdfs:range    :CSDevType ;
                       owl:inverseOf :hasChannel.

    :hasDirectionType  rdfs:domain   :CSDevChannel
```

Figure 1: Sample statements from K-500 Control System Ontology in Turtle syntax.

Benefits over Conventional Data Models

First, ontological store shares the same qualities that graph data model has, what allows expressing complex relations in a natural way. It becomes possible via describing *metadata* – data about data, or features that are used for structuring, searching and identifying objects in the knowledge base. Ontological model is very flexible, it allows to render multiple levels of abstraction and help avoid impedance mismatch [7] problem.

What makes ontological approach superior to simple graph database, is that knowledge base is an intellectual system. It does not manipulate with raw unstructured data, but handles facts and statements about the domain of knowledge. Not only do we describe objects and relations in the knowledge base, but also restrictions on these relations and rules for automatic generation of new facts. This provides possibilities for semantic analysis methods, such as consistency verification or inference of assumed knowledge. These powerful instruments are provided by the component of the knowledge base called *Reasoning engine* and are crucial, according to the requirements for consistency and integrity of data in the storage.

Another benefit is additivity of RDF data. Information stored in the knowledge base may come from various sources; in case of control system domain these are configuration files and schemes, which are frequently updated on experimental complexes. With triplet data model, merging process is very simple and new knowledge about the domain, which defines it more precisely, can be easily added at any time, not necessary at the design stage. This makes ontologies stable to changes in data or metadata, in contrast to rigid relational databases, which have restricted structure of tables.

INFRASTRUCTURE OF FACILITY KNOWLEDGE BASE

Apart from formalising knowledge about the domain of discourse, we should provide necessary conditions for making use of it. Every application that works with semantic data is called *Semantic Web application* [8] and should contain three components (Fig. 2).

Figure 2: Components of Semantic Web application and their functions.

RDF Store

RDF store, also called *triple store,* is the database that integrates metadata of the domain with actual data and sometimes with knowledge from external ontologies. It is responsible for retrieving knowledge base statements from various sources and storing them persistently. The choice of the triple store defines volume of data that can be stored, application operating speed and scalability. There are multiple relational and graph RDF store implementations from different vendors.

Reasoning Engine

The process of *inference*, or *reasoning* over stated information, means broadening stored knowledge with the facts that can be deduced according to the rules defined for this domain. For example, we can define that every instance of class “Terminal” is also a “Computer”, restrict number and types of ports it may have, put constraints on the way it can be connected to the local network, etc. Such rules help avoid duplicated facts in storage and allow to reveal inaccuracies before false data is inserted.

Reasoning engines either execute reasoning task being triggered by users’ queries (“just in time inferencing”) or materialize the inferred statements in the RDF-store (“cached inferencing”). [9]

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPL05

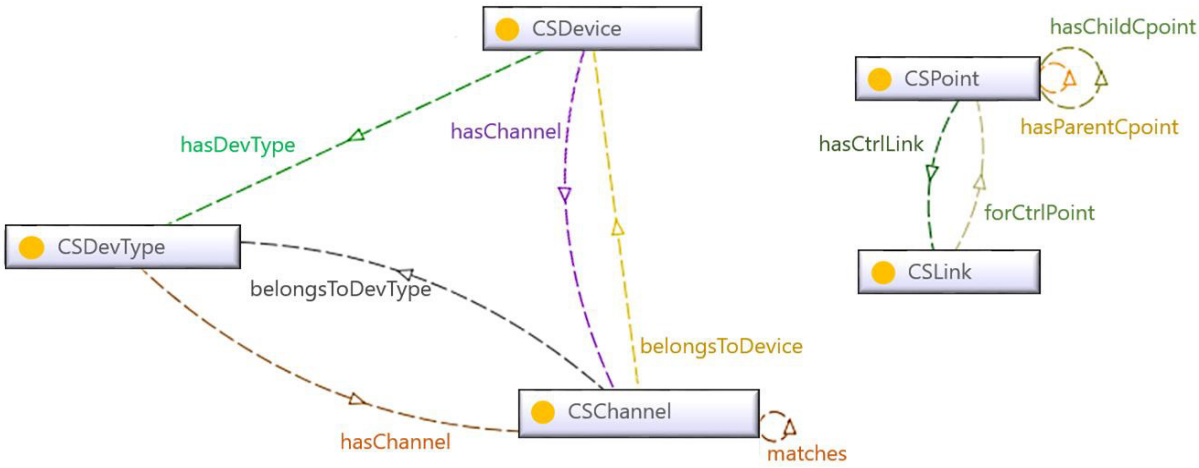

Figure 3: Scheme of metamodel of K-500 control system ontology.

Query Processor

Unlike relational databases with SQL language, RDF store uses *SPARQL Protocol And RDF Query Language* (SPARQL) [10]. RDF data can be accessed by performing SELECT, ASK, CONSTRUCT, DESCRIBE queries with extensive vocabulary of modifies. Query Processor reads this queries, mines the knowledge graph to find triplets coinciding with the given template and provides the results in CSV, XML or JSON formats.

KNOWLEDGE BASE DEVELOPMENT PROCESS

Designing Metamodel

The first step in the development of configuration data knowledge base was to design the ontological metamodel, which describes all concepts and relations existing in the domain of K-500 control system (Fig. 3).

With the use of *Protégé* [11] – open-source ontology editor and framework for building intelligent systems, we created ontology that describes K-500 control system classes and their characteristics, relations between instances of these classes and rules that should be used to infer knowledge. The designed metamodel forms the basis for the knowledge base since it contains all domain semantics.

Populating Ontological Model

Information about control system devices comes from configurational scripts located on server. We retrieve this data with the developed parsing instrument: for each record in configuration file it creates an instance of ontology class associated with this record. The parser serializes information about objects and relations between them by forming triplet statements. All in all, ontology model required 11043 axioms describing K-500 control system configurations and they were serialized in RDF format. *Apache Jena* framework was applied as an instrument for editing ontologies from Java application.

Choosing Triple Store

The next step was the deployment of the database, filling it with statements about K-500 control system domain and providing interfaces for working with stored knowledge. After the analysis of existing triplet store solutions, we chose *AllegroGraph* [12] semantic store. It is a commercial graph database, but it provides off-the-shelf solution with basic reasoning opportunities, APIs on multiple languages and interactive user interfaces, which was ideal for the first prototype of our knowledge system.

Although *AllegroGraph* showed convincing results of using semantic modelling approach, later we changed it for *RDF4J* triple store. This platform and Java framework for parsing, storing, inferencing and querying over RDF-data has been known as *Sesame* [13]. It is an open-source product and shows higher performance [14] on executing SPARQL queries.

Interacting with Users

One of the aims we want to reach with the developed platform is coming up with the solution of convenient way of presenting configuration data, which is crucial for control system users, but is a tough task due to the volume of data and the complexity of its structure.

In the first prototype of the knowledge base we implemented and tested the solution offered by *AllegroGraph* – interactive browser (Fig. 4) that provides table and graph view options, SPARQL queries execution and downloading results in different formats. However, it is not customizable and such a low-level presentation of data will be hardly used by anyone except ontology administrators.

We came to the conclusion that user interfaces cannot be made universal for all ontologies. They should rest upon domain knowledge peculiarities and solve actual problems that users of control system face. Therefore, we designed K500ConfigPortal web application based on *Flask* framework, which contains business logic of semantic data use and provides user interfaces that help solve the problem of documenting and visualizing configuration data.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPL05

Figure 5 displays overall architecture of the developed platform. K-500 Configuration Portal communicates with storage via REST API, sends SPARQL queries to it, processes RDF data results and presents them on web pages. The developed web application currently provides the following functional features:

- Browsing objects of control system, their attributes and relations in a wiki-styled navigator;
- Performing lexical search with semantic filters to find object matching particular parameters;
- Executing SPARQL queries and publishing their results as separate web pages (visualizations).

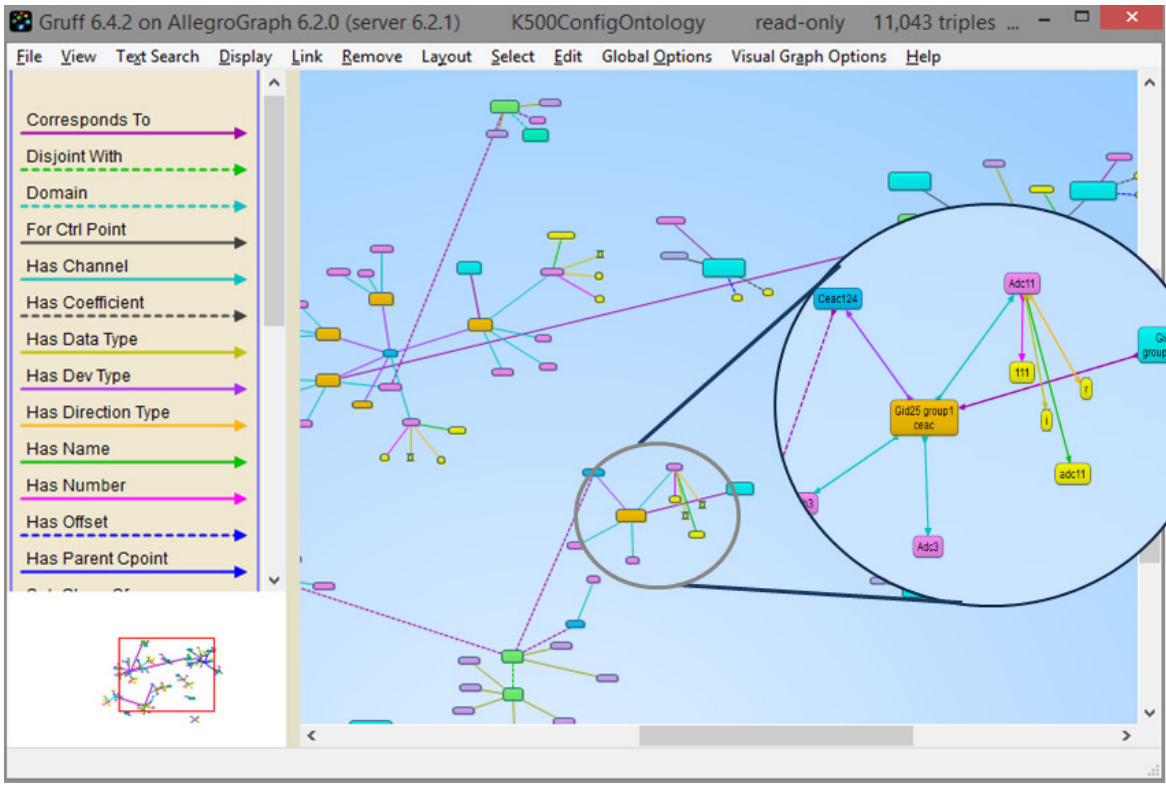

Figure 4: User interface for browsing ontology stored in AllegroGraph database.

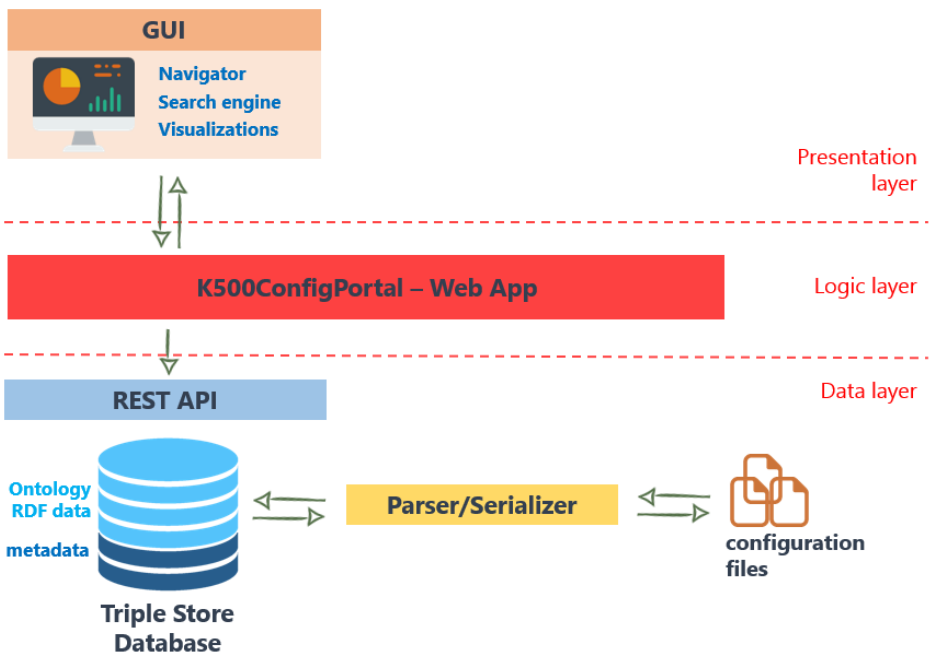

Figure 5: Architecture of the semantic platform for K-500 facility control system.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPL05

CONCLUSION

The semantic platform for storing and manipulating facility configuration data is currently under development. The analysis done during the work showed that ontological approach allows us to solve the majority of problems defined for configuration data storage.

The centralized knowledge base for K-500 facility control system that we developed represents facility elements and complex relations between them in a natural way. Furthermore, it enables knowledge inference and data consistency validation. The developed web application for access, navigation and visualisation of data is built as semantic service and solves the problem of configuration data documentation.

Although the development of the semantic platform is a time-consuming process, it is justified for large experimental complexes automatization. In perspective, this knowledge base will be enlarged by information about other aspects of facility and will provide a wide range of functional possibilities.

REFERENCES

[1] D. Berkaev *et al.,* "VEPP-5 Injection Complex: two colliders operation experience" in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, paper WEPIK026, pp. 2982-2984.

[2] D. Bolkhovityanov, P. Cheblakov and F. Emanov "CXv4, a Modular Control System", in *Proc. ICALEPCS'15*, Melbourne, Australia, October 2015, paper WEPGF093, pp. 915-918.

[3] P. Cheblakov, S. Karnaev and O. Khudayberdieva "Application of PyCDB for K-500 beam transfer line" in *Proc. ICALEPCS'15*. Melbourne, Australia, October 2015, paper WEPGF095, pp. 923-925.

[4] N. Nilsson "Logic and artificial intelligence", *Artificial Intelligence,* vol. 47, pp. 34-40, Jan. 1991.

[5] N. Guarino, D. Oberle and S. Staab "What is an Ontology?", in *Handbook on Ontologies,* S. Staab Ed. Springer-Verlag Berlin Heidelberg, 2009, pp. 1-17.

[6] OWL 2 Web Ontology Language. W3C recommendation 2012, https://www.w3.org/TR/owl2-overview/.

[7] I. Robinson, J. Webber and E. Eifrem, Graph Databases, 2nd ed. O'Reilly Media, Inc., Sebastopol, USA. 2015, pp.11-27.

[8] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez. Semantic Web Programming. Wiley Publishing, Inc. Indianapolis, USA, 2009, pp. 263-273.

[9] D. Allemang and J. Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL – 2nd ed. Elsevier Inc., Waltham, USA, 2011, pp. 117-123.

[10] SPARQL Query Language for RDF. W3C recommendation 2008, http://www.w3.org/TR/rdf-sparql-query/.

[11] Protege, http://protege.stanford.edu/.

[12] Introduction to AllegroGraph database, https://franz.com/agraph/support/documentation/current/agraph-introduction.html/.

[13] J. Broekstra, A. Kampman and F. Harmelen "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema" in *Proc. ISWC '02*, London, UK, 2002, pp. 54–68.

[14] K. Rohloff *et al*. "An evaluation of triple-store technologies for large data stores" in *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, Springer, Berlin, Heidelberg, 2007, pp. 1105-1114.