Semantics and Reasoning for Control Application Engineering Models

David Hästbacka and Seppo Kuikka

Tampere University of Technology,
Department of Automation Science and Engineering,
P.O. Box 692, FI-33101 Tampere, Finland
{david.hastbacka,seppo.kuikka}@tut.fi
http://www.tut.fi/ase

Abstract. Development of advanced systems requires new methods to improve quality and efficiency of engineering processes, and to assist management of complex models encompassing different engineering disciplines. Methods such as model-driven development and domain-specific modeling facilitate development from this perspective but reduce interoperability and other prospects of rationalizing processes, on the other hand. An approach applying OWL semantics and reasoning to models is presented with examples to support industrial control application engineering. Using the methods, generalized classifications are inferred from instance models and combined with generic engineering knowledge maintained in ontologies. Reasoning allows identifying assemblies and structures outside the scope of traditional modeling to detect flaws and error-prone designs. The results indicate that OWL semantics and reasoning can be used as a supplement furthering typical development practices.

Keywords: control application engineering, software models, semantic web, owl, rules, reasoning.

1 Introduction

Advanced control systems and applications are essential in monitoring and controlling industrial processes and manufacturing operations. The increase in the level of automation and intelligent features as well as the requirements on performance, reliability and safety of these systems has resulted in engineering challenges. There is a demand for new methods to improve development and engineering in order to address these requirements, and also to improve efficiency of engineering processes and reduce the total costs of system development.

In system development, models are used to develop, document and communicate engineering artefacts. For advanced systems the models easily become large including tens of thousands of engineering objects. Especially when people from different engineering disciplines use these models to communicate and collaborate, and as input for automatic imports and transformations there is a risk for error when potential problems and dependencies in models are not detected.

In our previous work, model-driven development (MDD) of industrial process control applications has been studied with emphasis on development processes and domain-specific modeling constructs [1]. The developed engineering process along with UML Automation Profile (UML AP) concepts enables the designer to focus on significant engineering challenges while much of the work between design phases is automated. Nevertheless, situations can be identified where the development process and the modeling foundation is not sufficient in providing the interoperability and support for further improving the engineering processes.

Interest in ontologies and related formalisms for industrial applications has increased during the last years [2]. The use of Semantic Web technologies can improve software engineering throughout the life-cycle by providing logic-based formalisms and semantics to concepts [3]. Despite many benefits it is problematic to shift to an entirely ontology-based modeling due to the differing nature of the approach. The varying granularity, freedom in expressivity, and the required complexity in modeling detailed semantics are examples of some of the impediments. Semantic Web technologies can regardless of the above mentioned be used as a supplement to MDD practices. For example, to provide added semantics and interoperability during engineering phases to improve understandability, knowledge management and reuse, automatic reasoning and classification, and even assisting in automatic transformations. Semantic descriptions generated from models could also be used for run-time operations [4][5].

From modeling perspective OWL does not allow syntactical enforcement of specific restrictions as opposed to typical modeling languages. However, this point of view can be neglected as the metamodel of the MDD approach typically caters for those aspects. The purpose of using semantic methods is to capture elements and features outside the scope of the metamodel. It is acknowledged that Object Constraint Language (OCL) can be used in many cases for defining constraints and rules to identify structures, related to the metamodel. OCL, however, is restricted to known types of the modeling language making it challenging for maintaining more advanced knowledge of a more generic nature.

This paper presents application of semantics and reasoning on MDD models in engineering of control applications using OWL 2 DL and SWRL. Knowledge management and reuse of engineering information and existing know-how is also discussed. Section 2 presents related work and background. The organization of information in engineering ontologies is presented in section 3 and OWL reasoning challenges for control application models are presented in section 4. The developed prototypes, examples and the results are discussed in section 5. Finally, section 6 contains the discussion and section 7 concludes the paper.

2 Related Work and Background

The complexity of consistency checking of UML class diagrams has been studied by using first-order predicate logic [6]. Description Logics based reasoning has been considered and applied to UML class diagrams to check for inconsistencies and redundancy [7] and the authors opinion is that state-of-the-art DL-based systems are ready to serve as a core reasoning engine in advanced case tools. Also a framework for integrated use of UML class-based models and OWL has been proposed [8]. Semantics in ontologies have been used to enhance modeling capabilities and transformations in MDD of service-based software [9].

The use of ontology-based query retrieval for reusing UML class diagram designs during development has also been studied [10]. The results highlight the importance of the domain ontology and the added semantics the ontologies bring. The adoption of OMG MDA principles to ontology development have been studied by [11] to facilitate transformations between different ontology languages.

Ways to combine ontologies with metamodeling have been studied by [12] and [13]. This paper enhances and deepens the discussion in [13] on the relationship between ontologies and meta-modeling, and the model-driven development paradigm by applying classification and reasoning to models in process control application development to support engineering tasks.

3 Control Application Engineering Knowledge

Knowledge management in engineering of control applications can be organized into three categories; domain knowledge, model instances, and use case specific knowledge, as presented in figure 1. Domain knowledge represents information related to the area of interest, e.g. a specific engineering discipline or a type of systems or devices, and knowledge and practices such as modeling languages and standards. From a modeling perspective a domain ontology (DO) provides the central building blocks and constructs used in the application domain but with additional semantic interoperability. The domain knowledge is of a static nature, i.e. the ontologies do not change very often. This type of information can be generated automatically from existing sources such as metamodels or standards, for instance. Domain knowledge can also be mined and the extraction of domain ontologies from engineering domain handbooks has been proposed [14].

UML AP modeling concepts [1] extend both UML and Systems Modeling Language (SysML), and hence there are three metamodels with corresponding domain ontologies defining the semantics and the relationships. In this case the ontologies are taxonomies that primarily reflect relationships in the sense of classification and generalization of concepts. In OWL a too detailed DO easily causes unwanted inferences if all property domains and ranges are considered.

Model instance knowledge represents models under development or being studied, i.e. model instances serialized as an individual or instance ontology (IO). A plain representation of the individuals that relies on the corresponding DO allows for the transformation between the modeling paradigms to remain sufficiently simple [13]. The IO contains the individual class types, the associated data properties, and structural ownership relations of the source instance model.

Use case specific knowledge refers to knowledge that can be described as information used in analysis and reasoning, e.g. rules on typical design issues and pattern-like structures, company specific conventions, and concerns where special attention is desired. This knowledge is typically of a generic nature and not restricted to a particular modeling or development method. In this sense, also mapping and alignment of different ontologies fall into this category when information from various sources needs to be combined. The ability to combine and reuse existing knowledge in different ontologies is also a justification for using ontologies to support development, e.g. applying generic engineering knowledge to the IO of a UML AP model while performing automatic structural analysis.

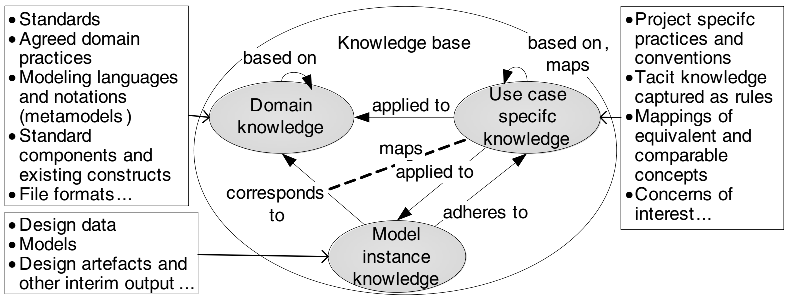
• Standards
• Agreed domain practices
• Modeling languages and notations (metamodels)
• Standard components and existing constructs
• File formats...
• Design data
• Models
• Design artefacts and other interim output ...
Knowledge base
based on
Domain knowledge
applied to
based on , maps
Use case specifc knowledge
maps
applied to
corresponds to
adheres to
Model instance knowledge
•Project specifc practices and conventions
•Tacit knowledge captured as rules
•Mappings of equivalent and comparable concepts
•Concerns of interest...

Fig. 1. The knowledge in engineering scenarios can be divided into three main categories: domain knowledge, model instance knowledge, and use case specific knowledge

4 Reasoning with OWL

OWL and SWRL are based on an open world assumption (OWA) meaning that anything that is not stated is unknown and cannot be used to deduce negation as a failure, for instance. As OWL DL (and SWRL) is based on description logics it supports only monotonic inference. Altering a fact based on some condition is not possible and requires an additional layer, e.g. program code, to be implemented. Also domain and range conditions are not constraints in the sense that they are checked for consistency. In reasoning they are used as axioms to infer further knowledge which can easily lead to unexpected effects e.g. in classification.

The impacts of open world semantics can be limited with techniques to close the world or restrict different possibilities. Traditional programming languages have often been used in combination with OWL to overcome these restrictions. Recently there has also been a RDF/SPARQL based proposal [15] for circumventing some of the limitations that is also applicable to OWL.

The management of truth in the knowledge base is of importance in scenarios where the knowledge is frequently updated, such as engineering environments. The challenge in forward chaining is that facts can be both explicit and implicit, and the same fact entailment can be based on a number of facts making the management difficult. On the other hand, inferring all the entailments each time can be too intensive from a performance point of view. This is worth noting as OWL DL, for instance, is computationally hard and in the NExpTime complexity class. Backward chaining is more attractive in the case of rapidly changing knowledge bases where inference is conducted only when needed and the additional entailments do not have to be stored and maintained.

Typically a lot of work is required backtracking different possibilities and reasoning is non-deterministic. Considering this, it can be argued that reasoning easily becomes computationally challenging even for simple-appearing problems. Although information can be stated more explicitly requiring less reasoning it is an important issue especially when integrating other information sources.

5 Applications in Control Application Engineering

The prototypes developed implement the concepts of knowledge management presented in section 3 with information distributed in separate OWL DL ontologies. The reasoning examples are performed mainly in Protege 4 but implementation as a Web Service in Java using OWLAPI has also been evaluated and proved working as well. The Pellet reasoner has been used in all of the examples both in Protege and the Java based OWLAPI implementation.

The SWRL based inferences are implemented DL Safe to retain decidability and are embedded in the engineering knowledge ontology (EKO) and the Mapping ontology (MO). UML AP model transformations to instance ontologies are performed using a refined version of the XSLT developed in [13]. Present are also the domain ontologies for UML AP, UML, and SysML metamodels.

5.1 Examples

Figure 2 illustrates some of the inferences using the developed ontologies for a subset of a control application model. UML AP model elements are connected with nested AutomationFunctionPort and InterlockPort elements that are linked using a UML Connector with ConnectorEnd sub elements identifying the Port elements. The structure is complicated and rules can be used to infer direct connections between model elements instead. The following MO rule considers UML AP elements (DO concepts and IO individuals) in the antecedent part and makes an inference of a simplified connection (EKO concepts) in the consequent.

```
AutomationFunctionPort(?prtA), AutomationFunctionPort(?prtB),
Connector(?cnn), ConnectorEnd(?cnnendA), ConnectorEnd(?cnnendB),
hasPart(?cnn, ?cnnendA), hasPart(?cnn, ?cnnendB),
hasLinkId(?cnnendA, ?idA), hasLinkId(?cnnendB, ?idB),
direction(?prtA, "out"^^string), id(?prtA, ?idA), id(?prtB, ?idB),
DifferentFrom (?prtA, ?prtB) -> hasPortConnectionOut(?prtA, ?prtB)
```

OWL allows declaring conditions for which new inferences can be made but the use of rules allows more powerful expressing of deductive reasoning than OWL alone. A similar example infers a tracedBy relation between Requirements and AutomationFunctions from a TraceRelation contained in the Requirement.

```
AutomationFunction(?af), Requirement(?r), TraceRelation(?tr),
hasPart(?r, ?tr), modelId(?tr, ?id), sourceDomainId(?af, ?id)
-> tracedBy(?af, ?r)
```

In addition to the inferred connections presented, also specific interlock connections are identified as well as relations between measurements that via a controller are connected to an actuator. For example the Primary and Secondary Controller inferences are reasoned based on an OWL class expression identifying a set point coming from another controller. The existence of cascade controllers in the organizing control loop also classifies it as a Cascade ControlLoop. In addition there is a Complete Connected ControlLoop inference that has identified all parts of the control loop to have the minimum amount of required connections.

If developing systems with safety requirements it could be required that all interlocks implemented must have a separate measurement for the interlock that is not used in the normal regulatory control of the actuator, for example. For this an inference can be made that identifies those actuators that have an interlock based on the same measurement that also the controller is utilizing.

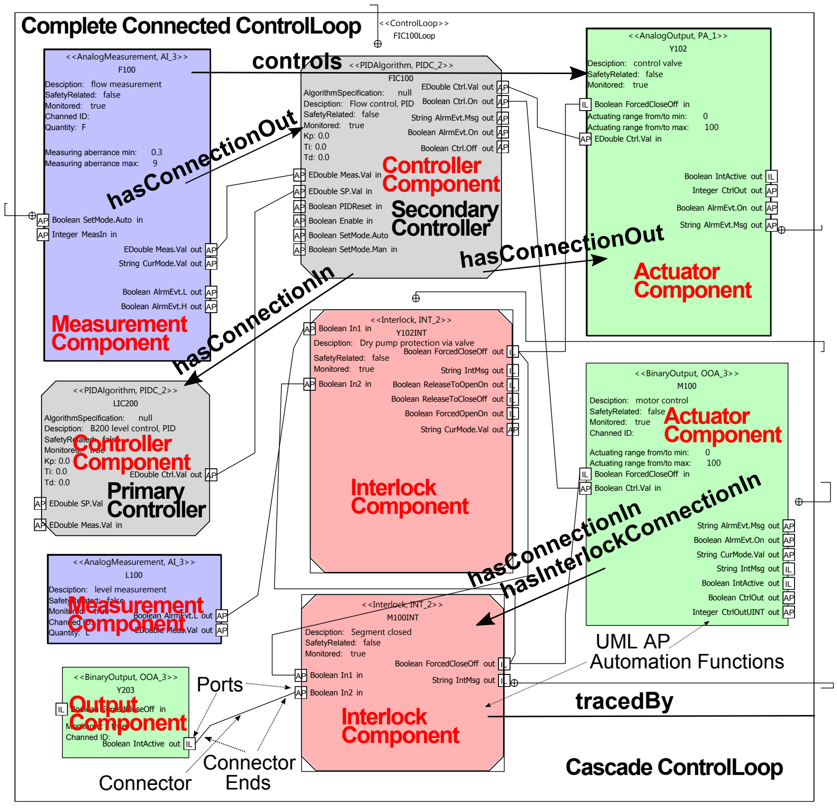

Fig. 2. Example of some of the generalized classifications and inferred connections for elements in a subset of a control application model. The red markings (lighter shade) represent the result of mapping assertions from UML AP ontology concepts to generic engineering concepts and the black markings further inferred facts about the model.

5.2 Results and Experiences

Classifications and inferences enable many use cases for the method to support engineering tasks and assist the developer by highlighting potential error-prone designs and structures, for example. It also enables the use of automatic checking of consistency and structures that the metamodel might not address.

Trace relations, for example, are used to relate requirements to functions in the model as a means to improve quality. Inferring the trace is of little use if broken or missing traces cannot be identified. Inferring this, the different interlock measurement or incompletely connected control loops, for example, is impossible using OWL mechanisms due to the OWA. In practice one can get around this if not developing applications that rely only on OWL. As the interesting concepts or structures could be defined as complements of those identifiable with OWL it was straightforward to make a complement using programming language constructs as a middle layer in the Java based prototype.

Rules were used to infer direct relationships between elements to simplify connections to the engineering knowledge. Rules were chosen because OWL does not allow mixing of object and data properties in chains. The rule based approach inferring simplified relationships based on linking sub elements proved to be challenging for models containing hundreds of Ports. In comparison, deducing the same connection assertions in the instance transformation phase, using XPath and XSLT, the typical ontology classification time was reduced to a tenth of the time required when the connection and interlock connection rules were included.

Using the reasoner also for the different individuals declaration of OWL was found decreasing performance significantly with more than one thousand individuals. An alternative approach using a functional data property to differentiate the individuals proved out more efficient.

A generic way to tackle large numbers of individuals is to perform reasoning only on a subset of the model and iterate the complete model one package at a time, for example. The feasibility of this, in general, depends on the source domain model structure of how sub models are connected. For rules performance a division of the reasoning tasks can be implemented by grouping rules to be executed according to the needs of each task. When there are a lot of rules involved it is, according to our experiences, usually quicker to run several smaller and even partially redundant reasoning tasks than a large one including all of the axioms.

Because UML AP is based on UML and SysML, and via its implementation on the Eclipse platform also on the Ecore metamodel, it is worth considering which of those equivalent ontologies are needed in reasoning. Unless utilization of knowledge in UML or SysML is not required in inferences, the DOs may be omitted from reasoning. For many purposes it is a feasible solution for UML AP because the model semantics are present in UML AP with its own class hierarchies, and the structures and semantics of UML and SysML are mainly utilized for the modeling tool support. Nevertheless, classification and operating with UML and SysML concepts is also possible for UML AP models if desired.

5.3 Additional Requirements for Design-Time Reasoning

Off-line analysis of models, e.g. in project repositories, can be implemented with less consideration on space and time complexity as it can be performed separately from design. In order to implement on-line reasoning properly, e.g. for integrated development environments (IDE), some additional issues have to be considered.

Reasoning performance is an important concern for support to be useful and assisting in development tasks. Therefore it is not practical to do all reasoning tasks as on-line background processing. A plausible approach could be to limit on-line reasoning only to a subset of selected objects and performing more extensive analysis less frequently, e.g. when saving or changing views.

Another issue to consider is the way the model instances in the IDE are transformed to the knowledge base. A practical transformation approach instead of the XSLT could be to use an incremental transformation that simultaneously maintains a semantic knowledge base of the model being developed.

6 Discussion

Interpreting models as ontologies enables various classifications and inferences to be performed. When knowledge about instances is combined with other information new inferences can be made to support development and give an indication about structures that designers should pay attention to, for example.

OWL provides capabilities for describing concepts beyond typical modeling languages. Additionally, OWL provides semantic interoperability which is an increasingly important feature both for networked engineering environments and the systems being designed. Using OWL, pattern-like structures can be identified and generic platform agnostic engineering knowledge can be applied and reused to classification and analysis of model instances.

In addition to practical issues which currently prevent adoption of pure OWL based engineering, it is also limited by some of its inherent design patterns originating from Description Logics. The OWA is a considerable restriction when reasoning on application models that reflect more a closed space. To close the assertional box an additional layer of program code or other techniques are required that e.g. operate on the less restricted OWL FULL or RDF level.

7 Conclusion

Development of advanced systems, such as industrial control applications, requires new methods to improve quality and efficiency of engineering, and to assist in handling of complex design models. Typical modeling methods, however, often fall short in interoperability, expressing additional knowledge, and reasoning on structures and features beyond single elements.

This paper presented application of OWL semantics and reasoning to models when developing control applications. Using the developed method, elements in models can be classified and generic engineering knowledge can be applied to detect inconsistencies and anomalies in model instances. According to our experiences OWL can be used as a supplement to MDD to provide interoperability and support in various engineering operations developing complex systems. The presented implementation and experiments are of an off-line nature but the general approach and some of the implementations, i.e. classifying a subset of a model, can also be adapted to on-line reasoning in an IDE as well.

References

1. Hästbacka, D., Vepsäläinen, T., Kuikka, S.: Model-driven development of industrial process control applications. Journal of Systems and Software 84(7), 1100–1113 (2011)
2. Breslin, J.G., O'Sullivan, D., Passant, A., Vasiliu, L.: Semantic web computing in industry. Computers in Industry 61(8), 729–741 (2010)
3. Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), Athens, USA (November 2006)
4. Albert, M., Cabot, J., Gómez, C., Pelechano, V.: Generating operation specifications from uml class diagrams: A model transformation approach. Data & Knowledge Engineering 70(4), 365–389 (2011)
5. Serral, E., Valderas, P., Pelechano, V.: Towards the model driven development of context-aware pervasive systems. Pervasive and Mobile Computing 6(2), 254–280 (2010)
6. Kaneiwa, K., Satoh, K.: On the complexities of consistency checking for restricted uml class diagrams. Theoretical Computer Science 411(2), 301–323 (2010)
7. Berardi, D., Calvanese, D., Giacomo, G.D.: Reasoning on uml class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
8. Parreiras, F.S., Staab, S.: Using ontologies with uml class-based modeling: The twouse approach. Data & Knowledge Engineering 69(11), 1194 –1207 (2010), special issue on contribution of ontologies in designing advanced information systems
9. Claus, P.: Semantic model-driven architecting of service-based software systems. Information and Software Technology 49(8), 838–850 (2007)
10. Robles, K., Fraga, A., Morato, J., Llorens, J.: Towards an ontology-based retrieval of uml class diagrams. Information and Software Technology 54(1), 72–86 (2012)
11. Cranefield, S., Pan, J.: Bridging the gap between the model-driven architecture and ontology engineering. International Journal of Human-Computer Studies 65(7), 595–609 (2007)
12. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. Journal of Systems and Software 84(2), 301–313 (2011)
13. Hästbacka, D., Kuikka, S.: Bridging uml profile based models and owl ontologies in model-driven development — industrial control application. In: International Joint Workshop on Information Value Management, Future Trends of Model-Driven Development, Recent Trends in SOA Based Information Systems and Modelling and Simulation, Verification and Validation, pp. 13–23
14. Hsieh, S.H., Lin, H.T., Chi, N.W., Chou, K.W., Lin, K.Y.: Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks. Advanced Engineering Informatics 25(2), 288–296 (2011)
15. Knublauch, H., Hendler, J.A., Idehen, K.: Spin - overview and motivation. Technical report (2011)