

Software Engineering Ontology for Software Engineering Knowledge Management in Multi-site Software Development Environment

Pornpit Wongthongtham¹, Elizabeth Chang¹, Ian Sommerville²

¹Digital Ecosystems and Business Intelligence Institute, Curtin University, Australia
 {Pornpit.Wongthongtham, Elizabeth.Chang}@cbs.curtin.edu.au

²School of Computer Science, St Andrews University, UK
 Ian@software-engin.com

Recently, there has been an explosion of interest in ontologies as artefacts to represent human knowledge and as a critical component in several applications. One unique area of research is the software engineering ontology. The software engineering ontology defines common sharable software engineering knowledge including particular project information. Software engineering ontology typically provides software engineering concepts – what they are, how they are related, and can be related to one another – for representing and communicating over software engineering knowledge and project information. These concepts facilitate common understanding of software engineering project information to all the distributed members of a development team in a multi-site development environment. Reaching a consensus of understanding is of benefit in a distributed multi-site software development environment. Software engineering knowledge is represented in the software engineering ontology whose instantiations are undergoing evolution. Software engineering ontology instantiations signify project information which is shared and has evolved to reflect project development, changes in software requirements or in the design process, to incorporate additional functionality to systems or to allow incremental improvement, etc.

The software engineering ontology contains abstractions of the software engineering domain concepts and instantiations. There are two types of the abstraction which are the generic software engineering and the specific software engineering. The abstraction of the generic one represents the concepts that are common to a whole set of software engineering concepts, while the abstraction of the specific one represents the set of software engineering concepts that are specifically used for some categories of particular projects. The instantiations, also known as population, are simply the project data. The abstraction of the specific software engineering ontology has its instantiations utilised for storing data instances of the projects. Each abstraction can have multiple instantiations in different circumstances of projects. The corresponding concrete data instances are stored as instantiations.

Software engineering ontology instantiations are derived as a result of populating software engineering project information and are referred to as ontology instances of software engineering ontology classes. The transformation process is usually accomplished by mapping various project data and project agreement to the concepts defined in the software engineering ontology. An example of transformation is given here. Figure 1 shows an example UML class diagram that will be transformed into the class diagram ontology model as instance knowledge.

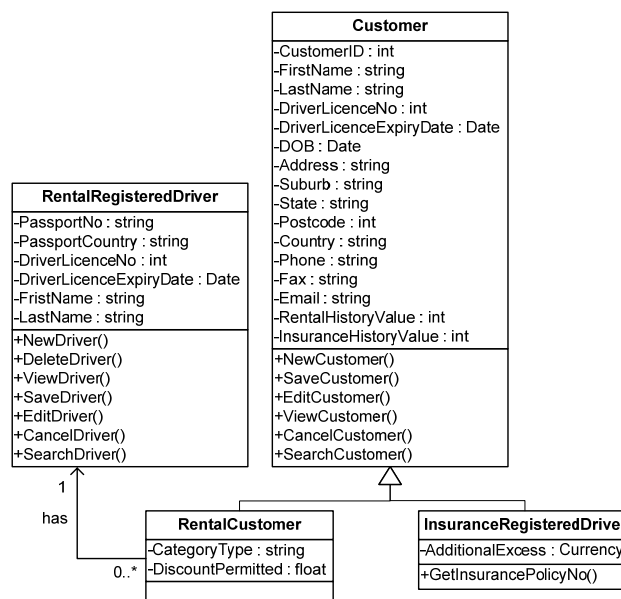


Fig. 1 An example of an UML class diagram

As from figure 1, UML classes Customer, RentalCustomer, InsuranceRegisteredDriver, and RentalRegisteredDriver apply as instances of the ontology concept Class in class diagrams ontology. Explicit domain knowledge from concept Class elicit that class consists of its properties, its operations and its relationships. This is by referring respectively, in the class diagrams ontology, to relations Class_Attribute, Class_Operation, and association ontology class ClassRelationship. The concept Class instance Customer has relation has_Attribute with concept ClassAttribute instances CustomerID, FirstName, LastName, DriverLicenceNo, etc. For example, the concept instance DriverLicenceNo has relations Class_Attribute_Datatype with xsd:string of 'Integer' and has relations Class_Attribute_Visibility with xsd:string of 'Private'. For a particular UML class Customer, operation NewCustomer() applies as an instance of concept ClassOperation in the class diagrams ontology model. The concept Class instance Customer has relation Class_Operation with concept ClassOperation instance NewCustomer. The concept instance NewCustomer has relations Class_Operation_Visibility with xsd:string of 'Public'.

For the particular class diagram shown in figure 1, the concepts of generalisation relationship and association relationship are applied. An instance of concept ClassGeneralisation has relations Related_Object_Class_Component with concept Class instances RentalCustomer and InsuranceRegisteredDriver and has relations Relating_Object_Class_Component with concept Class instance Customer. Instance of concept ClassAssociation has relations Related_Object_Class_Component with concept Class instance RentalRegisteredDriver, has relations Relating_Object_Class_Component with concept Class instance RentalCustomer, has relations Related_Cardinality with xsd:String of '1', has relations Relating_Cardinality with xsd:String of '0..*', and has relations Related_Role_Name with xsd:String of 'has'. These are shown respectively in figures 2, 3, and 4.

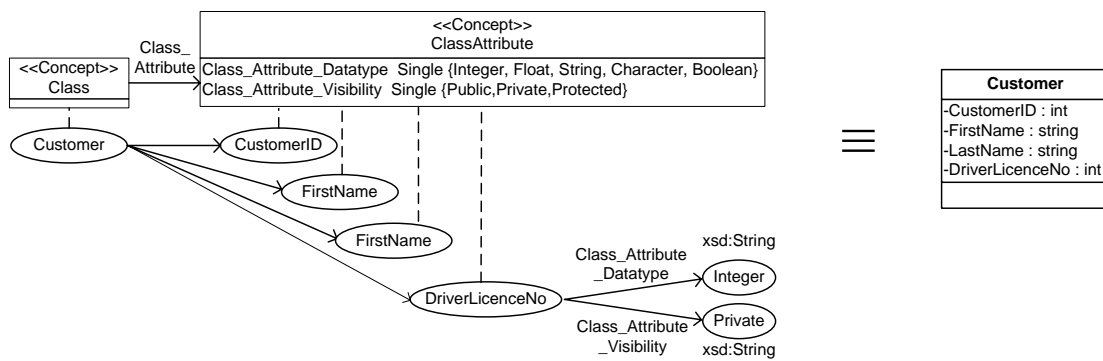


Fig. 2 Transformation of UML class Customer and its attributes to class diagrams ontology

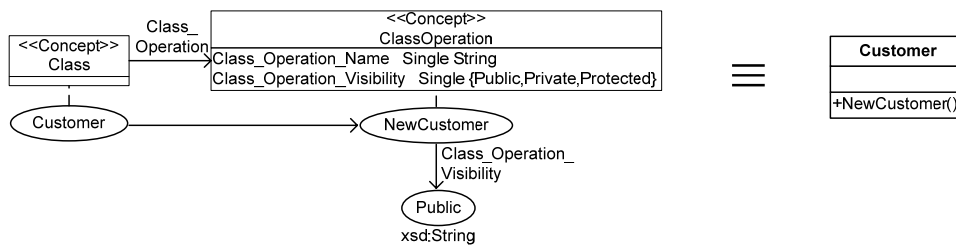


Fig. 3 Transformation of UML class Customer and its operation to class diagrams ontology

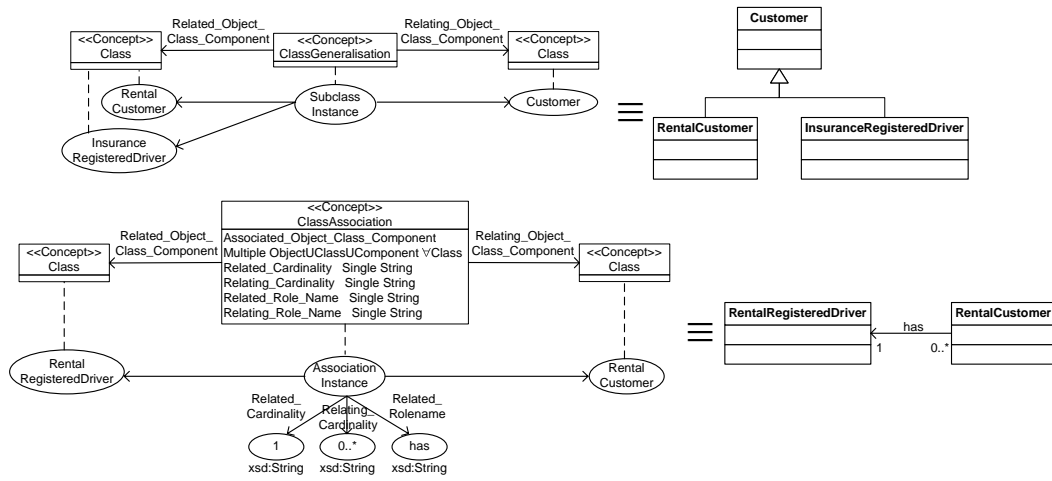


Fig. 4 Transformation of generalisation and association relationships to class diagrams ontology

Problem encountered in the multi-site environment is all about remote communication. The text transcription is difficult because work is carried out in an environment where development teams are geographically distributed and team members are involved in many projects simultaneously. It is a typical means of global communication which is, however, neither efficient nor sufficient for a multi-site environment. Figure 5 shows such an example of the text transcription that, in a multi-site environment, makes less of an impression.

I am struggling to understand why we need it. I think the system will be simpler for people to understand if we deleted the insurance registered driver. My reasons for this are that the insurance registered driver is a sub type of the customer. This means that for every insurance registered driver object there must be a corresponding customer object. However, in the customer object we store values like customer type, insurance history value and rental history value. It does not make sense to have these values for the insurance registered driver. I also think people will be confused because we have the rental registered driver as an association with the rental customer (which is a sub type of the customer) but the insurance registered driver is a sub type of the customer.

Fig. 5 An example of text transcription which is not efficient or sufficient for multi-site communication

With ontology-based software engineering, the software engineering terms can be parsed with software engineering ontology concepts and can recall the necessary details and relevant information. We see from figure 5 that it involves the terms of class (class insurance registered driver, class customer, and class rental customer), subclass (sub type), property (property customer type, property insurance history value, and property rental history value), and object (object insurance registered driver and object customer). Terms class, subclass, property, and object apply respectively to the concepts of class, generalisation relationship, class property, and class object in the software engineering ontology. By specifying ontology class instances, relevant information of those instances can be discovered dynamically and automatically.

This is carried out by referring to software engineering ontology which asserts that concept class has its semantic of containing attributes, operations, and relationships holding among other classes. Automatically drawing out details facilitates others' greater understanding of the content, thereby reducing misunderstanding, and eliminating ambiguity.

In conclusion, the software engineering ontology facilitates collaboration of remote teams in multi-site distributed software development. We have explored software engineering knowledge formed in the software engineering ontology. We have analysed instantiations in the software engineering ontology through the examples.