Towards Behaviorally Enriched Semantic RESTful Interfaces Using OWL2

Irum Rauf and Ivan Porres

Åbo Akademi University, Dept. of Information Technologies,Turku, Finland
{irauf,iporres}@abo.fi

Abstract. In this paper, we discuss how to represent behavioral semantic RESTful interfaces using OWL2. The conceptual and behavioral model of the service interface are designed using UML and then given semantic representation in OWL2. These semantic RESTful interfaces carry information that can be used to validate web service and can also be published for automated discovery and composition processes. Different ontology reasoners can be used to validate the consistency of these semantic RESTful interfaces.

1 Introduction

Representational State Transfer (REST)[2] has become a popular approach for developing web services. Usually, RESTful services offer a simple interface to create, retrieve, update and delete (CRUD) resources. However, it is possible to create RESTful web services(WS) with complex operations beyond basic CRUD.

Designing and publishing such RESTful WS is not a trivial task. Previously, we highlighted this need and presented a design approach to create and publish behavioral RESTful WS interface [4]. The use of semantic technology with web services facilitates automation, discovery and composition of web services. We are interested in using semantic technology for representing behavioral RESTful WS interfaces that can be used with different ontology tools and offer automated solutions for RESTful interfaces.

In this work we use OWL2 to design behavioral semantic RESTful web service interface that contain application states. The service is represented as conceptual model(CM) and behavioral model(BM) using UML[6] and then semantically represented with OWL2 [1]. The semantic representation of RESTful WS interface can be validated for its consistency and satisfiability using ontology reasoners.

2 Behavioral RESTful Interfaces

We use a simple example of a RESTful hotel booking WS to demonstrate our work. The service takes payment from the customer and books a room in the hotel.

In our Hotel Booking (HB) RESTful example, we take *booking* as a central element since the service books a room. All other resources are linked to it and

S. Auer, O. Diaz, and G.A. Papadopoulos (Eds.): ICWE 2011, LNCS 6757, pp. 407–410, 2011.

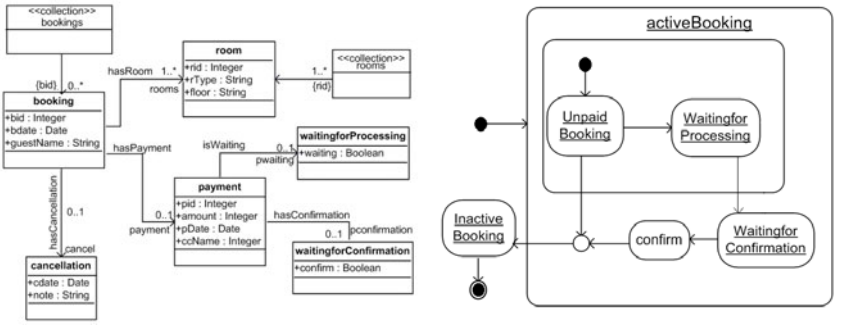

Fig. 1. (Left) CM for HB Web Service. (Right) BM of HB Web Service

are navigated through this central resource. The resources identified initially for HB service are shown in Figure 1(left). Figure 1(right) shows lifecycle of the *booking* resource with states representing different application states. A *booking* resource is created as an *activeBooking* and can have many other substates when it is active. A booking can be made inactive only if it is not waiting for the payment confirmation. Also, only an *inactiveBooking* can be deleted. For the detail implementation of the example, readers are referred to [5].

3 Behavioral RESTFful Interfaces in OWL2

Each resource in CM is shown as a class in ontology. The ontology classes are connected via object properties that represent the association between resources, i.e. connectivity. Every class is declared disjoint with the other defining the fact that each class represents different objects. The listing below shows the excerpt of OWL 2 representation of CM in Figure 1(left). We do not semantically represent collection resources, class attribute and cardinalities as they would clutter interface ontology with details that are not involved in validating the interface. These details, however, can be added to make the semantic interface descriptive if required.

```
Declaration( Class ( booking))
Declaration( Class ( room))
Declaration( Class ( payment))
...
Declaration( ObjectProperty ( containsBooking))
Declaration( ObjectProperty ( hasRoom))
Declaration( ObjectProperty ( hasCancellation))
...
ObjectPropertyDomain( hasRoom booking)
ObjectPropertyRange( hasRoom room)
...
DisjointClasses( booking room payment cancellation
  waitPayment confirmPayment)
```

The BM of HB RESTful web service in Figure 1(right) results in emergence of many new concepts in our ontology. Each state represents a piece of information and can be exposed as a resource (ontology class).

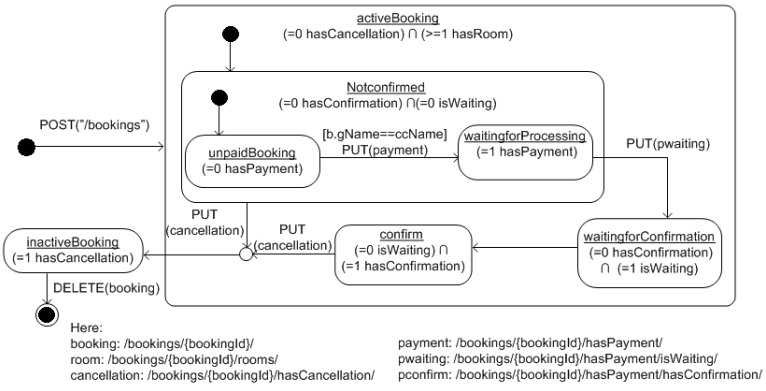

Fig. 2. Behavioral Model for HB RESTful WS with state invariants

```
Declaration( Class ( activeBooking))
Declaration( Class ( Notconfirmed))
Declaration( Class ( unpaidBooking))
.....
```

Next, the state hierarchy, i.e., class hierarchy of resources is defined that specify which resource is inherited from which resource. That is if state s_1 is a substate of s_2, OWL2 class c_1 represents UML state s_1 and OWL2 class c_2 represents UML state s_2 then c_1 is a subclass of c_2.

```
SubClassOf( unpaidBooking Notconfirmed)
SubClassOf( waitingforProcessing Notconfirmed)
....
```

The states at the same level of class hierarchy are mutually exclusive. The `DisjointClasses` axiom allows us to state this property of the state machine.

```
DisjointClasses( activeBooking inactiveBooking)
DisjointClasses( Notconfirmed confirm)
....
```

4 Behavioral Interfaces and Ontology Reasoners

The main design decision for BM is concerning the allowed methods. We allow four HTTP methods, i.e., GET, PUT, POST and DELETE, on the set of resources in state machine. GET is idempotent and is invoked to get the current state of the resource. PUT, POST and DELETE have side-effects and can trigger a transition from one application state to another. Figure 2 shows the behavioral model of our example annotated with service requests and state invariants. These state invariants link CM and BM of the interface. Each application state is defined by a state invariant, i.e., a boolean expression that links resources together to define an application state and is represented in OWL2 as:

```
SubClassOf (activeBooking
        ObjectIntersectionOf (ObjectExactCardinality(0
            hasCancellation cancel) ObjectExactCardinality(1 hasRoom
            room)))
SubClassOf (Notconfirmed
        ObjectIntersectionOf (ObjectExactCardinality(0 isWaiting
            waitPayment) ObjectExactCardinality(0 hasConfirmation
            confirmPayment)))
...
```

For detailed description on how state invariants are inferred for different types of states in UML protocol state machine, readers are referred to [3].

This OWL2 representation of state invariants provide semantic representation of applications states of a RESTful web service. Different OWL2 reasoners can be used to validate the consistency and satisfiability of these semantic behavioral interfaces. We must ensure that ontology describing CM should be consistent and all its resources should be satisfiable. Otherwise there cannot actually exist resources that conform to the interface described by our service. Each state invariant must be satisfiable and given two states that are at the same hierarchy level, their invariants should be mutually exclusive. For the classes to be satisfiable in the reasoner means that there exist resources that satisfy these application states.

5 Conclusion

In this paper, we discuss an approach to semantically represent behaviorally enriched RESTful WS interface using OWL2. These semantic behavioral RESTful interfaces can be published for automated discovery and composition and can also be used with ontology reasoners to discover inconsistencies at the design time.

References

1. Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3 Recommendation, (http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/)
2. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)
3. Porres, I., Rauf, I.: From uml protocol statemachines to class contracts. In: Procceedings of the International Conference on Software Test, Verification and Validation (ICST 2010) (2010)
4. Porres, I., Rauf, I.: Modeling behavioral restful web service interfaces in uml. In: In the Proceedings of 26th Annual ACM Symposium on Applied Computing Track on Service Oriented Architectures and Programming (SAC 2011) (2011)
5. Rauf, I., Porres, I.: Beyond CRUD-REST: From Research to Practice, 1st edn. Springer, Heidelberg (2011)
6. OMG UML. 2.2 Superstructure Specification. OMG ed (2009), http://www.omg.org/spec/UML/2.2/