RESOURCE ORIENTED MODELLING
Describing Restful Web Services using Collaboration Diagrams

Areeb Alowisheq, David E. Millard and Thanassis Tiropanis
School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K.

Keywords: Resource Oriented Architecture, REST, UML collaboration diagrams, Semantic Web Services.

Abstract: The popularity of Resource Oriented and RESTful Web Services is increasing rapidly. In these, resources are key actors in the interfaces, in contrast to other approaches where services, messages or objects are. This distinctive feature necessitates a new approach for modelling RESTful interfaces providing a more intuitive mapping from model to implementation than could be achieved with non-resource methods. With this objective we propose an approach to describe Resource Oriented and RESTful Web Services based on UML collaboration diagrams. Then use it to model scenarios from several problem domains, arguing that Resource Oriented and RESTful Web Services can be used in systems which go beyond ad-hoc integration. Using the scenarios we demonstrate how the approach is useful for: eliciting domain ontologies; identifying recurring patterns; and capturing static and dynamic aspects of the interface.

1 INTRODUCTION

The increasing popularity of RESTful Web Services is based on a number of factors like: being light-weight, providing easy accessibility, and being resource-oriented and declarative (Zhao and Doshi, 2009). This creates a demand for a modelling technique to abstract design from implementation. There are several approaches for modelling RESTful and Resource-Oriented (RO) Web Services, based on process calculus and related methods; however we adopt a more familiar approach (using UML) focusing on resources, which contributed to the success of RO and RESTful Web Services.

The advantages of Resource-Oriented Modelling lie from it being a more natural way to represent REST and ROA solutions, allowing designs to be easily mapped to solutions. It provides a simple mechanism for eliciting domain ontologies and captures dynamic and static aspects of the interface, it enables us to identify patterns across different domains. In section 2 existing approaches for RESTful and RO modelling are discussed. Section 3 discusses REST, ROA and our modelling approach. In section 4, scenarios are modelled from different domains. Section 5 will discuss its advantages.

2 RELATED WORK

Several approaches are proposed to model RESTful or ROA Web Services. Overdick (2007) shows how ROA is modelled using π-calculus, and since there is a mapping from Business Process Modelling Notation (BPMN) to π-calculus, then business processes can be modelled in ROA. Zhou and Doshi (2009) categorised WS into three types; they described them with ontology and rules and provided a framework for composing those services based on situation calculus. In work by (2010) resources were modelled in triple spaces, and a process calculus method was used to describe resource composition. These approaches overlook the REST constraint: hypermedia as the engine of application state, meaning that servers guide clients' transitions. They require formal descriptions which is not intuitive to most developers. In our work we use UML collaboration diagrams.

3 RO MODELLING

3.1 REST and ROA

Despite REST's popularity, it is misunderstood and oversimplified. Fielding, an author of the HTTP and

Alowisheq A., E. Millard D. and Tiropanis T..
RESOURCE ORIENTED MODELLING - Describing Restful Web Services using Collaboration Diagrams.
DOI: 10.5220/0003518401130118
In *Proceedings of the International Conference on e-Business* (ICE-B-2011), pages 113-118
ISBN: 978-989-8425-70-6

URI web standards, introduced the REST architecture style in his PhD dissertation (Fielding, 2000). The aim of his thesis was to realise the architectural aspects that made the Web successful as a scalable network-based hypermedia system. The constraints are: a client-server architecture, statelessness, cache, uniform interface, layered, and code on demand. These provide scalability, portability, simple replication of servers, reliability, efficiency, visibility, decoupling, and reusability. Developers welcomed REST because it provided a uniform interface without imposing additional layers. Many service providers like Google, Yahoo and Amazon started offering RESTful Web Services; however this rapid uptake came with the cost of not adhering to REST. The so-called RESTful Web Services violate two of REST's constraints: the uniform interface and statelessness. The need for a guide on how to design RESTful Web Services was met by Richardson and Ruby (2007), who focus on Resource-Oriented Architecture (ROA). The main idea in ROA is for the server to identify the resources and provide a uniform interface for them, through which a client can create, read, update and delete the resources. These actions are mapped respectively to the HTTP methods, POST, GET, PUT and DELETE. Fielding criticised ROA for not focusing on the hypermedia constraint. This entails using media types to specify not only the resources, but also the controls that indicate which actions can be performed. An example in HTML, the `<form>`, indicates GET or POST. The difficulty in discussing RESTful Web Service solutions lies in the fact that existing Web Service existing Web Service representations focus on services or messages. In our work we have sought to develop a resource-oriented modelling approach using UML Collaboration Diagrams.

3.2 The UML Collaboration Diagrams for RO Modelling

The UML collaboration diagram is one of the UML interaction diagrams (Booch et al., 2005) and it shows the interaction between objects and their structural organisation. It can model static and dynamic aspects of the system. When building ROA and RESTful Web Service, we are creating an interface not a complete system; therefore our modelling approach focuses on the interface. The interface is formed by the resources that the server exposes to the client. In our modelling approach resources take the place of objects in collaboration diagrams. According to ROA, these resources have a uniform interface: they can be created, read, updated or deleted.

Sending a POST request to a factory resource, or a class in UML terms, creates a resource. Figure 2 describes a Web Service for ordering pizzas. The client reads the menu, and then submits its order.

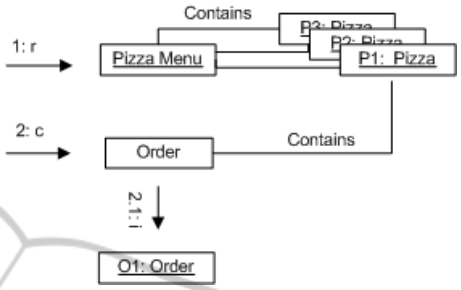

Figure 1: RO Diagram.

r, c and i on the messages respectively correspond to read, create and instantiate. The links labelled `Contains` are structural links showing how resources relate to each other.

4 RO MODELLING OF PROBLEM DOMAINS' SCENARIOS

We have chosen five scenarios each from a key problem domain. These domains are: Web mashups, Enterprise Services, Business to Business (B2B), Cloud Computing and Grid Computing. In each domain we present a scenario, and its RO modelling. Our intention is to provide evidence our technique works across a range of important domains, and then in Section 5 show how it facilitates their analysis.

4.1 Yahoo Pipes (Mashups)

Mashups combine APIs and data sources to form new applications and new data sources. This scenario is creating a mashup using Yahoo Pipes, an interactive web application that enables the creation and execution of mashups. A user can add widgets, such as data sources, and filters to merge the data.

A user has built a stock quote mashup using Yahoo Pipes(Donnelly, 2010), it displays last quotes and chart for stocks. He uses the widgets to retrieve original stock data from a .csv file at Yahoo Finance downloads. Then he uses a widget to filter the stock file for stock quotes. To loop through the obtained data he uses a widget to display the results as a chart.

The generic scenario of building mashups using Yahoo Pipes is broken down to the following steps:

(1.) The client creates a mashup
(2.) The client creates widgets
(3.) The widget produces the results
(4.) The client reads the results

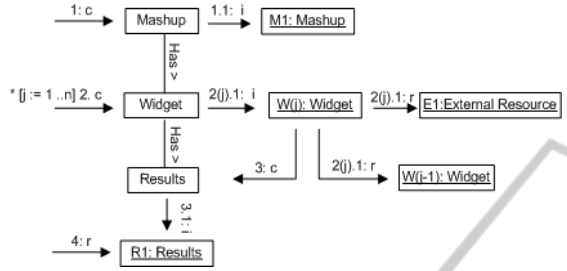

Figure 2: Modelling Mashups Creations with Yahoo Pipes.

4.2 City University (Enterprise Services)

Enterprise Services integrate different systems, whilst maintaining independent evolution of these components. The scenario chosen is an integration project from City University (2008) called Single Sourcing of Programme Data (SSPD). Information about the study programmes is used in different processes, however these operate independently this leads to inconsistencies in data and effort duplication.

SSPD is concerned with how programme information is created, updated and used, so that different processes could be facilitated and any inconsistencies resolved. It enables academic and administrative staff to maintain module and programme specifications and submit for approval.

This scenario can be decomposed into:

(1.) Academic Staff reads the programme info
(2.) Creates a modification
(3.) Can update it, when it is finished
(4.) It is approved by the Administrative staff
(5.) The programme info is updated
(6.) It can be read by interested processes

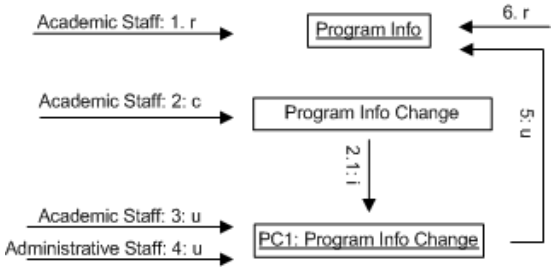

Figure 3: Modelling City University's SSPD.

4.3 Reverse Auctioning (B2B)

Business to Business services offer the ability to share information and performing transactions on the Web. The scenario modelled here is a reverse auctioning scenario from (Decker and Weske, 2007):

"A buyer (e.g., car manufacturer) uses reverse auctioning for procuring specially designed components. In order to get help with selecting the right suppliers and organizing and managing the auction, the buyer outsources these activities to an auctioning service. The auctioning service advertises the auction, before different suppliers can request the permission to participate in it. The suppliers determine the shipper that would deliver the components to the buyer or provide a list of shippers with different transport costs and quality levels, where the buyer can choose from. Once the auction has started, the suppliers can bid for the lowest price. At the end, the buyer selects the supplier according to the lowest bid. After the auction is over, the auctioning service is paid."

The scenario could be broken down into:

(1.) The buyer creates an auction
(2.) The buyer starts the auction
(3.) The suppliers place their bids
(4.) The buyer selects a bid
(5.) The buyer pays for the service
(6.) The buyer deletes the auction

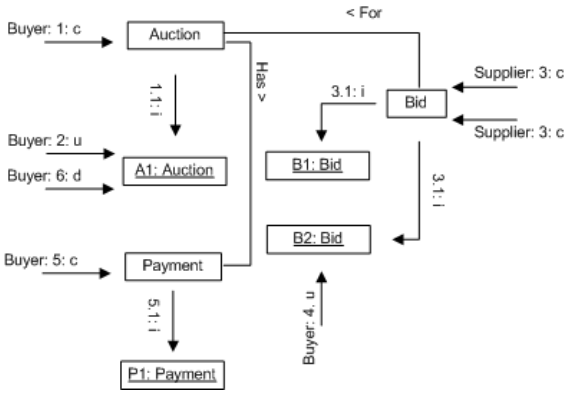

Figure 4: Modelling Reverse Auctioning.

4.4 TimesMachine (Cloud Computing)

Cloud computing offers software, platforms and infrastructures as services to clients. These are dynamically scalable to respond to high peak loads. The cloud computing scenario we chose is the New York Times project called TimesMachine (Klems et al., 2009), which aims to provide access to issues dating back to 1851, adding up to 11 million articles.

The team wanted to generate the PDF files from TIFF images. They decided to generate all the PDF files and serve them on request. The size of TIFF files was 4 Terabytes. So they used Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3). The TIFF files were uploaded to S3, they started a Hadoop cluster of 100 customized EC2 Machine Images. They transferred the conversion application. That resulted in the conversion to PDFs and storing the results to S3 taking 36 hours only.

The decomposition of the scenario:

(1.) Create the data items, upload the images
(2.) Create a Hadoop Cluster
(3.) Create an application and upload it
(4.) The application returns the results
(5.) The client reads the results

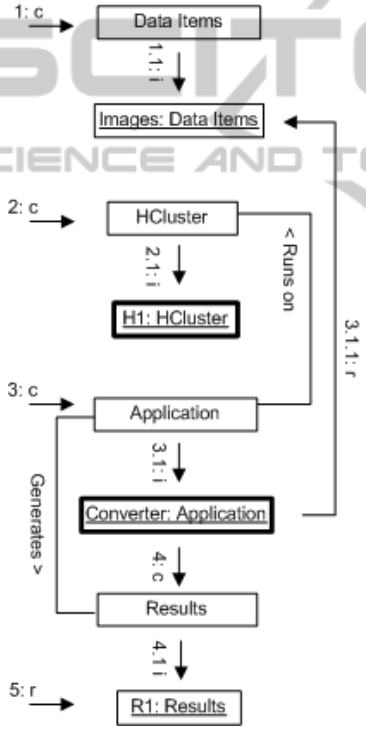

Figure 5: NYT Cloud Computing RO Model.

4.5 NEESgrid (Grid Computing)

Grid Computing is concerned with enabling the utilisation of distributed resources to provide a seamless platform for computational or data-intensive applications. This platform is used to enable remote collaboration and instrument sharing. NEESgrid is an NSF funded project to build a virtual laboratory for earthquake engineers. Using grid technologies enables remote access and control to observational sensors, experimental data, computational resources, and earthquake engineering control systems such as shake tables and reaction walls (Gullapalli et al., 2004).

Earthquake engineers wanted to study the effect of an earthquake on different types of substances and structures, these different structures and their shake tables are distributed across a number of labs, the aim was to coordinate these experiments with computer simulations. So the Multi-site Online Simulation Test (MOST) was devised to test and illustrate this capability using the NEESgrid system. MOST coupled physical experiments testing the effect of an earthquake on the interior of a multi-story building at 3 different sites each testing a part of the structure. MOST linked the physical experiments at the University of Illinois at Urbana-Champaign (UIUC) and at the University of Colorado, Boulder (CU) with a numerical simulation at the National Centre for Supercomputing Applications (NCSA). A simulation coordinator coordinates the overall experiment.

The scenario consists of the following steps:

(1.) Create experiments and the simulation
(2.) Create an experiment coordinator
(3.) The coordinator starts the experiments
(4.) The coordinator retrieves experiment results
(5.) The coordinator aggregates the results
(6.) The results are read

5 ADVANTAGES OF RO MODELLING

5.1 Eliciting Domain Ontologies

Semantic Web Service approaches such as SAWSDL (Farrell and Lausen, 2007), and OWL-S (Martin et al., 2004) require domain ontologies. The structural view that RO models offer can be used to elicit domain ontologies. By mapping the resource factories to classes, resources to objects and the links into relationships, the structure of the domain ontology can be elicited, what remains is to add the data properties. We can use this simple mapping to create the basis of an ontology in OWL (Bechhofer et al., 2004) for the scenario 4.3:

```
:Auction    a owl:Class.
:Bid        a owl:Class;
:Payment    a owl:Class.
:For        a owl:ObjectProperty;
rdfs:domain :Bid; rdfs:range :Auction.
:Has        a owl:ObjectProperty;
rdfs:domain :Auction; rdfs:range   :Payment.
```

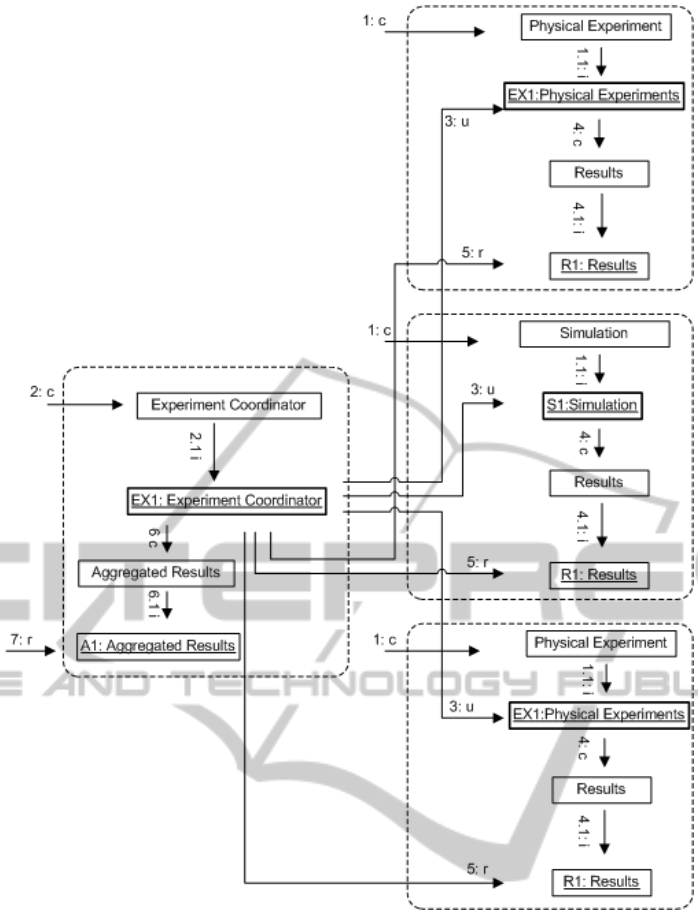

Figure 6: NEESgrid Experiment RO Model.

5.2 Modelling Static and Dynamic Aspects

This is a result of being based on UML collaboration diagrams. The static aspect of RO models informs developers on the resource type and the relationships between them from the client's point of view; in other words, the domain model. The dynamic aspect is shown by the messages showing the control flow: how the server needs to guide clients to achieve the functionality described, and what "next state" options the server should provide.

5.3 Identifying Recurring Patterns

RO models aid in identifying recurring patterns. Some we know from other software engineering areas. For example:

Factory: the factory is a well-known pattern that appears several times in all of the scenarios. In it a given object creates and initialises new objects.

Returning Results: This is where a resource creates results for a client to read. This appears in steps 3 and 4 in Figure 2, and steps 4 and 5 in Figure 5.

Controller: this occurs in Figure 6, where a resource updates several resources.

Identifying patterns can aid in providing RO solutions when modelling systems, and also in designing code generation tools for patterns, making development faster and less error-prone.

6 CONCLUSIONS AND FUTURE WORK

We introduced an RO modelling approach for modelling RESTful and RO Web Services. We used RO models to describe Web Services in five different problem domains. The approach models structural and behavioural aspects of the Web Service. The structural aspect can be used to elicit domain ontologies. Moreover RO models can be used to describe recurring patterns. Further work will be done to identify recurring patterns from the

RESTful and RO perspective; this will help in providing solutions to common problems and in informing design decisions for standards and platforms, which will emerge in this dynamic area.

REFERENCES

Bechhofer, S., Harmelen, F. V., Hendler, J., horrocks, I., Mcguinness, D. L., Patel-schneider, P. F. & Stein, L. A. 2004. OWL Web Ontology Language Reference. In: DEAN, M. & SCHREIBER, G. (eds.). W3C Recommendation, *World Wide Web Consortium* (W3C).

Booch, G., Rumbaugh, J. & Jacobson, I. 2005. Unified Modeling Language User Guide, Addison-Wesley Professional.

City University. 2008. Introducing SOA at City University, *City University*, London.

Decker, G. & Weske, M. 2007. Behavioral consistency for B2B process integration. *Advanced Information Systems Engineering Proceedings*, 4495, 81-95.

Donnelly, P. 2010. Yahoo Finance Stock Quote Watch List Feed [Online]. Yahoo. Available: http://pipes.yahoo.com/31337/watchlist [Accessed 26/02/2010 2010].

Farrell, J. & Lausen, H. 2007. Semantic Annotations for WSDL and XML Schema. *W3C Recommendation, World Wide Web Consortium* (W3C).

Fielding, R. T. 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, *University of California*.

Gullapalli, S., Dyke, S., Hubbard, P., Marcusiu, D., Pearlman, L. & Severance, C. Year. Showcasing the features and capabilities of NEESgrid: A grid based system for the earthquake engineering domain. In: the *13th IEEE International Symposium on High Performance Distributed Computing*, 4-6 June 2004 2004 Honolulu, Hawaii USA. 268-269.

Hernandez, A. G. & Garcia, M. N. M. Year. A Formal Definition of RESTful Semantic Web Services. In: *First International Workshop on RESTful Design* (WS-REST 2010), 2010 Raleigh, North Carolina. 39-45.

Klems, M., Nimis, J. & Tai, S. 2009. Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. Designing E-Business Systems, 22, 110-123.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcllraith, S., Narayanan, S., Paulocci, M., Parsia, B., Payne, T. R., Sirin, E., Srinivasan, N. & Sycara, K. 2004. OWL-S: Semantic Markup for Web Services. W3C Member Submission, World Wide Web Consortium (W3C).

Overdick, H. Year. The Resource-Oriented Architecture. In: the IEEE Congress on Services, 7-11 July 2008 2007 Hawaii, USA. 340-347.

Richardson, L. & Ruby, S. 2007. RESTful Web Services, O'Reilly Media.

Zhao, H. & Doshi, P. Year. Towards Automated RESTful Web Service Composition. In: *Proceedings of the 2009 IEEE International Conference on Web Services, 2009*. 1586928: IEEE Computer Society, 189-196.