Building Pattern with Open Web APIs in E-Governance

Biplav Srivastava, Jim Laredo, Shao Chun Li (Jery), Florian Pinel, Daniel Williams, Fang Quan Xie (Fred), Chih C Yuan (Paul)*, Blaine Dolph*

*IBM and IBM Research

Abstract

In this paper, we describe our experience in building a pattern around Open 311 APIs and working with it in a mobile application. With tens of cities around the world supporting Open 311, the pattern can scale to support building applications with them seamlessly. We find that patterns are indeed a promising direction for building (mobile and web) similar applications at scale.

Introduction

A quiet revolution is sweeping cities around the world in e-governance (S-CUBE 2010). First, they are putting more and more of their data available in public domain (India 2013, London 2013). Second, they are following common interfaces like 311 using open web standards (HTTP, REST, JSON/ XML) to expose their data. However, the ultimate success of these efforts is not in how much data is made available but how effectively they get consumed for society's benefit. The promise of open data trend is that developers and users can use the public data seamlessly to create value-added applications and insights, which will improve economic activity and reduce corruption, thus doubly re-invigorating the city's economy.

In practice, the number of applications being built and made available is a miniscule fraction of the number of datasets available. For example, in India (India 2013) has been publishing data since Dec 2012. Although the number of datasets is in thousands (7008 on 15 April 2014), the number of applications is in tens (25 on 15 April 2014). Many of the datasets come with API interface, which has been supported since March 2014.

To quicken and ease development of value-added applications, API patterns (Agarwal et al 2008) come in. They are high-level workflows about how APIs can be used together which the developer expected to be commonly reused and thus created. A prospective user (developer) can select the pattern from a catalog, map its service declarations to concrete API instantiations and automatically generate and host an application delivering the pattern's capabilities.

A Simple Open 311 Pattern

311 originally referred to a public telephone number in many cities in US and Canada to access non-emergency services like graffiti, garbage, down trees and abandoned car (Wikipedia 2014). It now also refers to a set of APIs called Open311 that provide access to 311 services (Open 311 2014) and are adopted by increasing number of cities around the world. The Open 311 APIs refer to six methods used to interact with these resources. We choose APIs from 4 cities in our demonstration: Chicago, Boston and Tucson in USA and Bonn in Germany.

On use-case, let us suppose that we have a concerned person, Mary, who sees a problem on the road like garbage pileup, and is concerned to get it removed. She will like to know if that has already been reported, and if not, report it.

When she is in her native city, Chicago in USA, she will like to know what services that city provides. When she is travelling to a new city, like Bonn in Germany, she is equally concerned and wants to get the matter reported.

We see that Mary wants to access a common capability across different data contexts, which is city in our case. This allows us to define a pattern, which is a workflow of an abstract application delivering the common capability, by removing low-level details and parameterizing the API calls. The pattern can be instantiated with any city's service instance API information to generate a customized application for it. The value of a pattern is thus that it:

- Standardizes the usage experience by promoting similar behavior (for users)
- Simplifies application development by templatizing API interactions (for developers)
- Serves as the organization's memory of the best-practices in developing a class-of-applications even when the specific APIs may not be relevant (for business)

With patterns available for Open 311, let us consider how Mary will use the pattern-generated applications. She obtains the mobile app from her favorite vendor, launches a sub-application for the capability by selecting an icon and picks the city she is in. She

- Sees a small set of categories (health, building, traffic, cityimage, others) around which all the city's services are grouped. The name, description and scope of a service may vary with city, so the category serves as an initial guide.
- She can look at a list of services and check out the agencies involved
  - If there has been a change in agency responsible or new services added for an agency, she can note that directly
- By choosing different cities and focusing on a category, she can do a high-level comparison of how different cities serve their citizens despite differences in their offered service names and scope.

Using Patterns to Generate Applications

In order to operationalize the pattern concept, one has to use a cloud-based platform. We now describe some of the key steps we took using an IBM cloud infrastructure, but similar process could be followed with any equivalent alternative like public offerings (Cloud 2014).

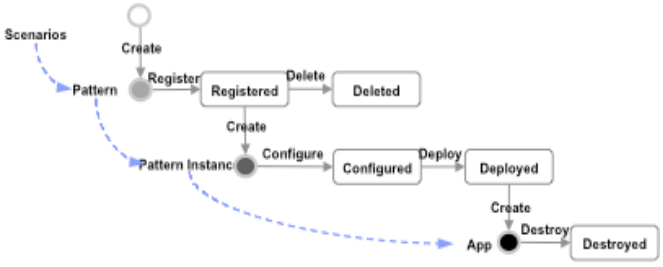
Pattern is a template of an application. In our case, it contains deployable assets, with configurable properties and bindings to service dependencies. It contains following key components:

- **Deployable Assets:** This can be J2EE WAR, PHP, Mobile App and similar distributions. These assets are automatically or manually converted and configured by abstracting of their Services API Dependencies and Configuration Parameters.
- **Services API Dependencies:** This describes the API dependencies information, including functional information including: API type, input/output, authentication method and types, and also non-functional information like Terms of Service (ToS) and Quality of Service (QoS) requirement. The dependencies are used by Assets or by runtime. In our pattern, dependencies to Open 311 end-points are specified.
- **Configurations:** They define the configuration parameter like UI style, Logic conditions, Runtime configurations. These parameter values will be either consumed by Assets or by runtime.
- **Provided Services APIs:** They describe the externalized (pattern instance's) API information provided after it gets deployed. This enables the pattern to provide value-based enhancement above and over what the dependent services provide. In our pattern, the (service) group information falls under this category.
- **Documents:** They are documents (PPT, DOC, HTML etc) with detailed guidance and introduction for instantiated (pattern) application.

In Figure 1, the process of registering and using (instantiating) the pattern in our environment is shown. The registration happens on a portal, which records the pattern information mentioned above. In parallel, for each city that the environment wants to support, a registration of the external city API is done exposing the details of its Open 311 end points.

A future user discovers the pattern on the portal based on scenario of interest and explores its configuration points. The portal also shows all the APIs that are registered on it and can be used to instantiate the pattern. The user selects the pattern, configures and automatically generates an instance, and deploys the instantiated pattern. The application is now available for use like any other web application. At a later stage, the deployed application can be destroyed as desired.

Figure 1: Registering and Using Patterns



Acknowledgements

We thank Jie Cui (Jessica), Arjun Natrajan, Xin Zhou (Cindy), Yew-huey Liu, Karthik Visweswariah, Sougata Mukherjea, Apurva Kumar and Maja Vukovic for discussions on various aspects of pattern mechanisms.

References

Wikipedia 2014. 3-1-1. At http://en.wikipedia.org/wiki/3-1-1, accessed 1 Dec 2013.

Open 311, 2014. Open 311. At http://open311.org/, accessed 1 Dec 2013.

S-CUBE, 2010. S-CUBE E-government Case Study, At http://scube-casestudies.ws.dei.polimi.it/index.php/E-Government_Case_Study, accessed 29 Jan 2014

V. Agarwal, G. Chafle, S. Mittal, B. Srivastava, Understanding Approaches for Web Service Composition and Execution , In ACM Compute 2008, Bangalore, India.

Cloud 2014. Cloud Foundry, At http://www.cloudfoundry.com , accessed 29 Jan 2014

India 2013. India's data portal. At http://data.gov.in . Accessed 30 June 2013.

London 2013. London's data portal. At http://data.london.gov.uk. Accessed 30 June 2013.