Open Government Data as a Service (GoDaaS): Big Data Platform for Mobile App Developers

Soheil Qanbari, Navid Rekabsaz and Schahram Dustdar
Distributed Systems Group, Vienna University of Technology, Vienna, Austria
Email: {qanbari, dustdar}@dsg.tuwien.ac.at, rekabsaz@ifs.tuwien.ac.at

***Abstract*—The next web of open and linked data leverages governmental data openness to improve the quality of social services. This data is a national asset. In this study, we elaborate on this emerging open government movement, together with the underlying data transparency to drive novel business models which utilize these assets under a functioning platform called *Open Government Data as a Service (GoDaaS)*. These business models actively engage civic-minded programmers in developing sustainable applications, contextualizing and utilizing the government open data resources. This leads to an expansive government marketplace, with many civic-minded developers might be new to doing business with the federal or state government. By means of a consultation service prototype, we provide development advices for programmers on how to work out the specific details of their applications business model. Having the business models in focus, this study also proposes a novel abstraction unit called *Gov. Data Compute Unit (DCU)*, so that governments are able to feed developers with formalized, structured and programmable data resource units rather than just data catalogs. Such DCUs enable developers to cope with an increasing heterogeneity of state government data sets, by providing a unified interface on top of diverse data schemata from various states.**

I. Introduction

Information theory begins with the representation, interpretation and transmission of patterns of data, i.e., patterns made up of different kinds of "*things*". We attach meanings to these patterns and call the result, "*information*". Patterns of data are mainly of interest when they are transmitted from a source to a receiver [1]. Along with this idea, governments are rich in such valuable information, as they must re-imagine their roles as an information provider. The Gov 2.0 movement welcomes civic intelligence with its data transparency, i.e., enabling governments to adapt to the ever-changing needs of their citizens. The *Data.gov* initiative, for instance, does not just catalog raw data; it takes this idea to a new level by providing a collection of open APIs[1] to government data. The rationale for *data.gov* is laid out in a reliable information infrastructure that "*exposes*" the underlying data to the public (i.e., online) at large. The Gov 2.0 platform model takes the further step of highlighting third-party *mobile applications* created by independent developers in a real "*AppStore*" like Washington, D.C.[2]. These repositories of data-driven apps can be open-sourced as a means for sharing best practices with other governmental bodies and cities. *Code for America*[3] is an instance of such open source collaborative business model.

Opening up and publishing raw data such as maps, employment statistics, weather surveys, agricultural statistics, and educational records, together with their associated APIs, while enforcing and respecting privacy policies of course, makes two things possible: (i) Enabling government as a platform and data infrastructure provider, who leverages GoDaaS to a new level of transparency to queryable sources of large amounts of underlying operational data. This leads to more control by citizens over their governments, as well as closer cooperation with them. (ii) Civic-minded programmers and the private sectors can build and deploy applications on government's data infrastructure to achieve optimal utilization of the data. This utilization is realized by creating new interfaces to government using the Open311[4] standard, developing *MobileApps* and offering new services in the *AppStore* as aided by government-provided GoDaaS data APIs. For instance, developers can register for a key at api.data.gov to access data offerings, via REST-full requests and returned responses in JSON or XML.

Moving forward, we frame the research question of GoDaaS: How does government become an open platform that allows civic-minded developers inside and outside government to develop practical applications? How to transform and expose government big data into tradable services? This study addresses these challenges by driving new classes of GoDaaS business models, in which stakeholders participate and share things. These business models evolve through interactions between government and its citizens, like a service provider enabling its user community. The increasing amounts of government data made available, coupled with unpredictable and ever-changing business requirements, triggered us to incorporate cloud service delivery models to define flexible and adaptive business models. To this end, our contribution is threefold: (i) Deriving seven business models for Government Open Data as a Service (GoDaaS) platform. (ii) Novel government data abstraction unit, called $Gov.\ DataComputeUnit(DCU)$ as a programming construct for developers. (iii) GoDaaS consultation service that generates a tailored and *application-specific* business model.

The paper continues with a survey on some contemporary related work on defining open government business models at section II. With some definitive clues on how the government data is currently traded, we define a new abstraction unit, called *Gov. Data Compute Unit (DCU)*, for government data at section III. This unifies our data resource trading unit within all proposed business models. To elicit and illustrate the need for GoDaaS from the requirements engineering perspective, section IV is devoted to the core stakeholders and their

[1] http://data.gov/developers/apis
[2] http://Apps.DC.gov
[3] http://codeforamerica.org/
[4] http://open311.org

978-1-4673-8103-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FiCloud.2015.34

relationships in GoDaaS ecosystem. Having the stakeholders' interest in the provision of government as GoDaaS platform, the focus is put on implementing this new model with the API requirements at section V. Subsequently, the API requirements associated with the corresponding stakeholders for GoDaaS are derived. Then, section VI presents a detailed view on GoDaaS platform architecture in support of our requirements. Having the architecture in place enables the proposed business models. As a proof of concept, three business models are described from developers view at section VII. In support of our model, we develop a primary GoDaaS consultation service for developers. The running prototype[5] is detailed at section VII-B. Finally, section VIII concludes the paper and presents an outlook on future research directions.

II. Related work

In relation to our work, there are some prominent studies on capturing business models for open government data. The European commission prepared a study on business models for Linked Open Government Data (LOGD) for the ISA[6] programme by PwC EU Services in late 2013 [2]. In this report, the authors provide a theoretical framework to analyze the LOGD. Fourteen entities who offered publishing, linking and accessing open government data as a service were selected as case studies for further analysis. The framework is structured according to the nine areas in the Business Model Canvas (BMC) [3]. There are considerable studies on how to provide government's data as *Linked Data*[7][4]. In [5], [6], [7], [8] and [9], the authors propose a semantic approach on attaching meaning to government data by applying ontologies to formally and semantically represent data.

Our paper outlines a set of abstractions for serving governmental data. Governments produce data that members of the public are entitled to access but format, size, and technology hurdles often prevent such access. GoDaaS is a proposal to mandate that all government data be made available in a form that can be accessed through a unifying set of programming abstractions.

III. Government Data Compute Units (DCU)

In the current Gov. 2.0 movement, governments are unable to feed developers with formalized and structured data. Government data is available in data catalogs or APIs, which are not published as a unit, but rather accessible on the data portal. For instance, the *data.gov* and its CKAN[8] API only contain meta-data about datasets. This meta-data includes URLs and descriptions of datasets, which is not handy for programmers. As a reaction to this complexity, we designed a new abstraction layer called *Data Compute Unit (DCU)*, that allows governments to express their data packages more structured and consistent, so that developers can utilize these packages by treating them as objects. DCU copes with an increasing heterogeneity of state government data sets, by providing a unified interface on top of diverse data schemata from various states. In this context, every state government

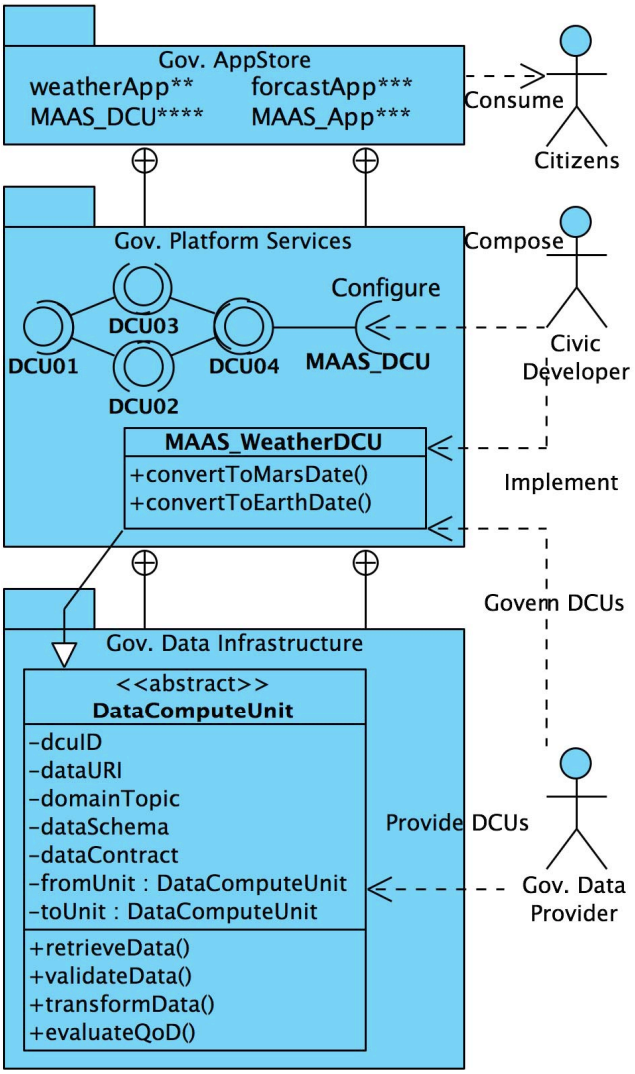

Fig. 1: GoDaaS Stakeholders together with their relationships.

provides its data in a unified interface of DCU. Then external systems, like SOAP services or even programmers are able to invoke an API from DCU library like *transformData()*.

We define a DCU abstraction as a GoDaaS platform programming construct, containing raw data and associated meta-data together with its utilization APIs. DCUs can be considered as a programming construct, like Classes, for developing data-intensive applications. Civic developers can compose, program and configure those DCUs to a package like Linked Compute Units[10] and expose them to services delivered in mobile apps to citizens. As shown in Fig. 1, DCUs are composed of two parts: The DCU meta data and set of APIs. The meta data provides: (i) a unique ID for future object referencing, (ii) an URI that identifies the data source, which is actually a URL that supports the data schema protocol for the retrieval purposes, (iii) a schema for an structured data extraction and loading, (iv) a contract where terms and usage licenses are detailed, and (v) two *fromUnit* plus *toUnit* elements to wire DCUs together for

[5]http://soheil4tuwien.github.io/GoDaaS/tool.html
[6]Interoperability Solutions for European Public Administrations (ISA)
[7]Tim Berners-Lee view: http://www.w3.org/DesignIssues/LinkedData.html
[8]http://ckan.org/

composition purposes. DCU provides programmatic access to the contents and meta-data of the government data repository and fosters re-use for programmers. We believe governments must provide underlying data resources as *DCUs*.

In this study, our resource unit granularity is a $Gov.\ DataComputeUnit$. Developers pull and program these units from government data infrastructure and expose DCUs as web services or mobile apps. For instance, NASA has developed an open source REST API, called MAAS[9], to provide information on the weather data being transmitted by the Curiosity Rover on Mars. We consider the MAAS API together with its meta data as *MAAS_WeatherDCU*. Developers can program and override this unit by building mobile weather apps or analytic applications to utilize the weather data for their research purposes. The MAAS API is available as an open source project under the Apache license[10].

Listing 1: Excerpt of DataComputeUnit Class Implementation

```
// Sample MAAS DCU class extending the abstract
// DataComputeUnit class

public class MAAS_WeatherDCU extends DataComputeUnit{
        private XML data;
        public MAAS_WeatherDCU(){
          dcuID = new GUID();
          dataURI =
          "http://marsweather.ingenology.com/v1/../";
          dataSchema = define XSD schema; }
        public XML retrieveData(){
                data = curl -X GET uri;
                return data; }
        public bool validateData(){
                return dataSchema.validate(data); }
        public JSON transformData(){
                return JSON.convert(data); }
        public float convertToFahrenheit(){
                int cels_temp = data.getTemprature();
                return (cels_temp * 9 / 5) + 32;} }
```

Listing 1 provides a closer touch of programming a DCU by the pseudo-code of some implemented procedures. The *MAAS_WeatherDCU* class implements abstract methods of *DataComputeUnit* class. In the class constructor, all *dcuID*, *dataURI*, and *dataSchema* attributes are initialized. The *dcuID* is of the GUID type, which makes it unique in the whole ecosystem. This unique value is useful for tracing and logging DCU objects. The *dataURI* points to the data source location for data retrieval, which can be an internal service. The *dataSchema* is defined as an XSD file in order to validate the quality of data variable structure. Invoking *retrieveData()*, fetches the data provided in URI address using curl command and stores it as XML format in the *data* variable. The *validateData()* procedure handles schema validation using XSD validation. Next, *transformData()* is a simple function for converting the *data* into JSON format. In the context of *MAASWeatherDCU*, the *convertToFahrenheit()* method fetches the stored temperature data and converts it to Fahrenheit. It implements a light-weight logic regarding to MAAS_WeatherDCU.

The GoDaaS architecture employs an enterprise service bus for its messaging, service integration and orchestration of processes. In Listing 2, we show the pseudo-code of core methods of Government Service Bus (GSB). Each *service* subscribes for a specific domain *topic*. The *GovServiceBus* discovers the queried *topic* in its governance repository and creates a new one if it is not already defined. Then, the *service* is added to the instantiated *topic*. The *publish()* function stores the DCU in the service bus *topic* queue. Using the *mediate()* function, the *dcu* objects stored in the *queue* are processed one by one. Based on the topic of the *dcu* object, subscribed *services* are discovered. Having these *services* identified, GSB checks whether the *service* has access to the *dcu* object. Finally, GSB delivers the *dcu* object by calling the *receive()* function of the *service*.

Listing 2: Excerpt of Gov. Service Bus Class Implementation

```
// Sample Government Service Bus (GSB) class
// dealing with DCU objects.

public class GovServiceBus {
public void subscribe(Service service,Topic topic){
          topic = find topic in the repository,
          create if not exist
          topic.addSubscriber(service); }
        public void publish(DataComputeUnit dcu){
                queue.push(dcu); }
        private void mediate(){
                DataComputeUnit dcu = queue.pull();
                for all dcu.topic.getSubscribers(){
                        authenticate(subscriber);
                        subscriber.receive(dcu);} } }
```

IV. Stakeholders in GoDaaS

To investigate and elicit the requirements, we classify core stakeholders into three main groups of *Government Data Provider*, *Civic-minded Developers* and *Citizens*. Their dependencies are illustrated in Fig. 1.

♢ *Government Data Provider*: This entity owns and provides the data. They provision an operating open DCU infrastructure together with the development platform.

♢ *Civic-minded Developers*: The civic developers implement the logic aspects of their *DCU-based* applications. They can program DCUs using the associated APIs for an intended behavior.

♢ *Citizens*: A citizen is the government service consumer. The government body opens up its data to its citizens for more transparency over their services. Citizens eventually may consume the data via mobile applications on the government *AppStore*.

V. API Requirements for GoDaaS

The *Eight Open Government Data Principles*[11] document outlines the key requirements for open government data. Embracing these eight principles, we delve into our perception of the APIs and derive technical requirements for the Cloud-based government platform (GoDaaS). In our approach, the government is to offer the GoDaaS platform and a set of APIs, so that developers consume those APIs for their application programming.

[9]http://marsweather.ingenology.com

[10]https://github.com/ingenology/mars_weather_api

[11]https://public.resource.org/8_principles.html

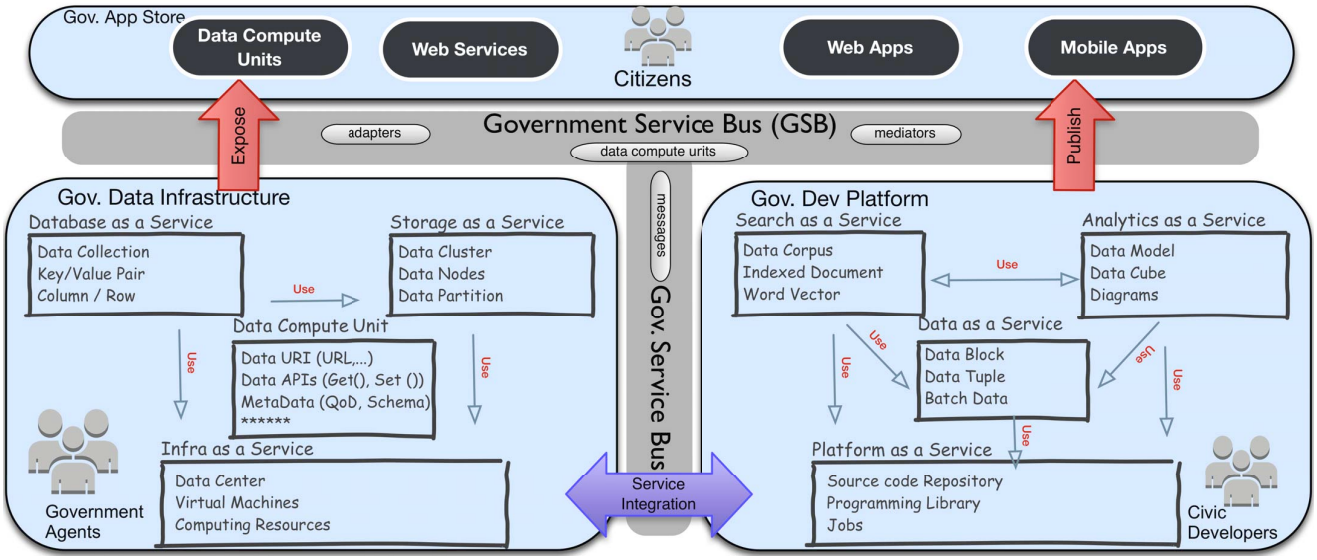

Fig. 2: GoDaaS system architecture in layers.

A. RQ1. End-to-End Governance Coverage throughout Data Compute Unit Life Cycle:

Governance enforces government policies, governing all aspects (e.g., manipulate, compose, expose, evaluate and control) of the DCUs throughout its life cycle. The stages of DCUs life cycle, through which published-data passes, can be sequenced as collection, processing, use, storage, application, provision and disposition.

B. RQ2. PaaS-enabled DevOps Integration for Government DCU-based App Programming:

This requirement deals with enabling the development environment, composition, adoption and use of DCU-based apps. As such, it incorporates full development life cycle tooling in support of application programming, debugging, testing, building and deploying processes. Government must provide the open DCU programming interfaces to its data.

C. RQ3. Discover, Subscribe and Provision Assets/Apps through a Government AppStore Interface:

Developers compose and program DCUs and expose them as services or mobile apps. Subsequently, the AppStore publishes these apps as marketable entities; then citizens are able to subscribe, keep them in use and pay as they go. This leads to more economies of scale for all parties involved. GoDaaS asset store not just enables the on-demand and automated provision of these apps, but also couples with clients' satisfaction and app recommendation service that meets citizens' requirements.

As the stakeholders and requirements relationship is listed in Table I, the need for asset/app store by each stakeholder stands out. In support of these requirements, we propose the GoDaaS platform. Next, we detail the GoDaaS layered architecture design, indicating the logical separation or division of components.

Stakeholders ↓ Requirements →	RQ1	RQ2	RQ3
Civic-minded Developer		•	•
Government (Federal, State)	•		•
Citizens (Crowd, Individual)			•

TABLE I: Stakeholders/Requirements relation in GoDaaS.

VI. GoDaaS Architecture

The impetus behind GoDaaS platform is how to adopt a data-driven cloud-enabled platform, which provides interfaces for developers and private sectors to develop their publicly available applications that expose the underlying data.

Fig. 2 illustrates the GoDaaS architecture, which encompasses in four major layers: (i) *Gov. Data Infrastructure*, (ii) *Gov. Development Platform*, (iii) *Gov. Service Bus* and (iv) *Gov. AppStore*. Using Gov. data infrastructure, the government incorporates cloud computing technologies to virtualize its data resources into *DCUs*. This enables elastic DCU provisioning, meaning that government is able to dynamically allocate DCUs to developers in an on-demand fashion. Conversely, the apps can pull DCUs by invoking storage or database services to complete a transaction. Along with this layer, the Gov. development platform layer is in place, where authorized developers can implement and deliver their applications on top of government data. This platform enables programming, composing and deployment of DCU-based applications. It is basically a government-class cloud platform, which employs data-intensive programming models supported by integrated development libraries and services like data analytics and search facilities, which will empower the development process. On top of these layers, the government AppStore stands out. The AppStore supports multi-tenancy, where identities like state governments, third parties and civic developers are authorized to deploy their applications for trading. Through governance services, the AppStore ensures that deployed apps

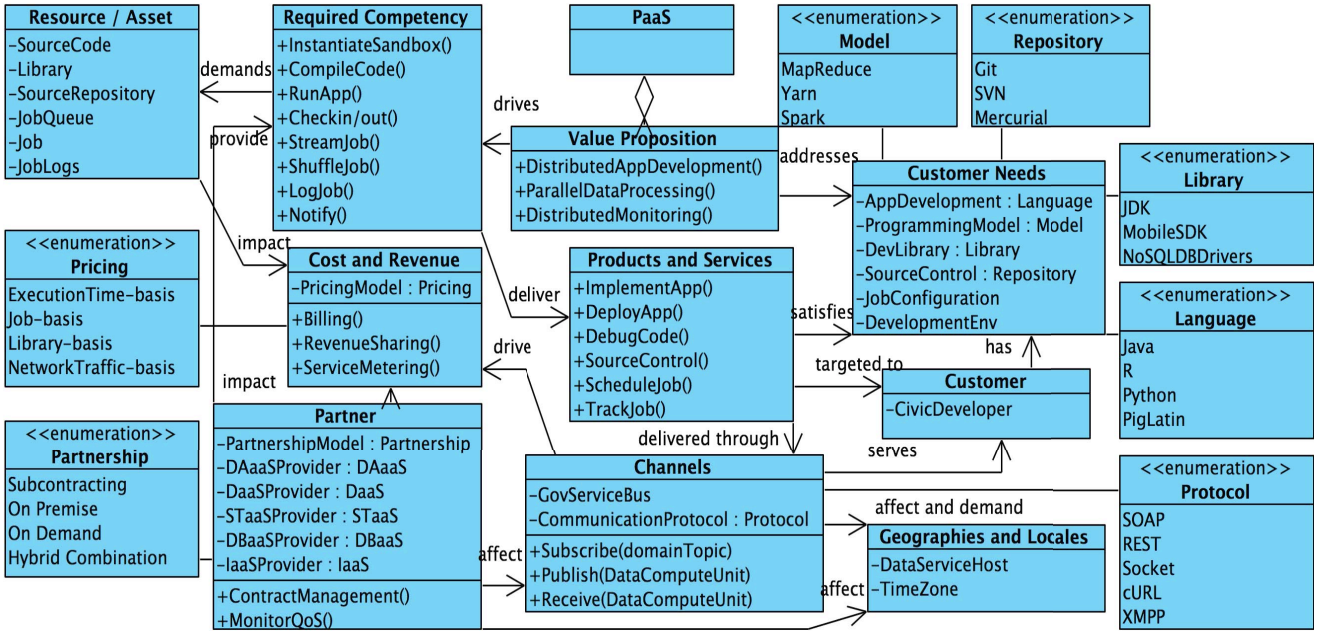

Fig. 3: GoDaaS Platform as a Service (PaaS) business motivation model class diagram.

comply with the government policies and regulations respectively. Moving forward, the monitoring, metering and billing services track the asset/app usage and debit citizens.

GoDaaS multi-layer architecture needs an integration enabler for its implementation and a uniform service delivery model. Government service bus (GSB) layer glues all the entities, agencies and services together through its messaging and queuing mechanisms. Together with the integration, GSB also provides data adapters to the DCUs registry, which is a point of access to all provided and customized DCUs by governments or developers. It acts as an intermediary component to accept and provision the requests to DCUs. Then it invokes the associated adapter to retrieve the DCU, validates and transforms it through data schema and forwards it to the developer, for instance.

In order to catalyze the adoption of cloud delivery models, we have come up with seven business models for the government open data. GoDaaS can include the following types of data-driven business models: *Data Infrastructure as a Service (IaaS)*, *Storage as a Service (STaaS)*, *Data as a Service (DaaS)*, *Database as a Service (DBaaS)*, *Platform as a Service (PaaS)*, *Search as a Service (SEaaS)* and *Data Analytics as a Service (DAaaS)*. All seven business models and their associated elements are consistently specified and modeled in details, mostly from the developer's view.

In this context, business modeling is driven by the value of API calls chain, as a sequence of service invocations to exchange DCUs among governments and developers. This value can be realized through developed mobile applications, that are consumed by citizens. For modeling purposes, we consider two main languages: the *Business Model Ontology* (BMO)[12] and the *OMG Business Motivation Model* (BMM)[13].

VII. GoDaaS Business Models

Due to space limitations, we only present *Database as a Service* and *Platform as a Service* business models in the paper. Since DCU is an abstraction layer between data corpus and developers, we integrate the DCU with several business models to help civic-minded programmers make better use of GoDaaS framework. All the interactions among business models in GoDaaS are based on the unified interface exposed by the DCU. We briefly recap the three models here, and refer the reader to the GoDaaS site[14] for further details of all seven implemented business models.

A. Gov. Database as a Service (DBaaS)

Government's parallel computation and distributed storage requirements to perform data-intensive analytic processes on their big data, motivate the use of elastic and scalable NoSQL databases such as Apache Hbase[15] or Casandra[16].

DBaaS allows federal governments to provide a single logical database to span geographically dispersed state governments' data centers, while maintaining intelligent load balancing. Storage load is provisioned on IaaS clouds in the form of virtual disks that can be attached and detached from running VM instances[11]. Dealing with such state-aware loads means enabling storage nodes to scale elastically, tightly interfaced with the STaaS and IaaS layers. Eventually the GoDaaS platform will expose database services as APIs to developers and will govern patterns of API usage.

[12]http://www.businessmodelgeneration.com/canvas

[13]http://www.omg.org/spec/BMM

[14]http://soheil4tuwien.github.io/GoDaaS/index.html

[15]HBase Homepage. http://hbase.apache.org

[16]Casandra Homepage. http://cassandra.apache.org

B. Gov. Platform as a Service

Government PaaS implements an application factory platform, enabling developers to *Code-on-Demand*[12]. The platform then incorporates the state or federal SLAs to ensure the government applications which are built through this process, follow a set of enforced guidelines. Data intensive government application development often involves large volumes and distributed data sets. Hence, the platform provides data-intensive parallel programming models to provide distributed control of code execution during job enactment. For instance, in the *MapReduce* programming model, data processing is broken up into distributed fine-grained *Map* and *Reduce* tasks to devise a parallel solution, performed by a *Master-Slave* design pattern.

As shown in Fig. 3 the government PaaS objective outlines the need for a *"Collaborative & Composable Platform"*, where the federal government would need to leverage "sharing" of DCU resources and fork open source applications among state governments for civic developers. This contributes to sharing best development practices and lessons learned from current celebrated *"government-ready"* mobile applications. Sharing such code artifacts leads to agility in the development cycle. The platform provides a collaborative development environment (using SVN, Git, etc.), accompanied with a set of programming APIs like *compileCode()*, *debugCode()*, etc., and some development libraries for seamless coding. The developers can import the available DCUs from the registry into their IDE, then program and compose configuration and code artifacts into a single development project and deployment archive called composite application.

As a prototype, we have developed a consultation service where developers can clarify their application specification and receive advice on their application's technical requirements for it's commercial possibilities. The aim of the tool is to design a tailored business model conforming to the needs of a new application. As a proof of concept, the service forms a business model for the DAaaS. The service is fed with a sequence diagram available in the referenced project repository, demonstrating the internal relations between the methods and attributes of the DAaaS business model. The service parses the sequence diagram and traces model element dependencies for new model generation. The tool is hosted and running on the GoDaaS site.[17]

VIII. Conclusion and Outlook

This paper presents a abstraction that combines government data URL and its associated interface called DCU to enable open government data as a service. The DCU also incorporates the government policy. Based on DCU, the authors expand the discussion to the opportunity of civil use of the government data and its potential advantages. In support of such abstraction layer, we present a reference architecture to enable the development of GoDaaS, that is, Government Data as a Service. The goal of the architecture is to provide a formal way for a given government to expose open-data, in a way developers with civic-minded ideas can easily reuse it instead of spreadsheets or PDF files as classically used. The key idea is to expose mechanisms that will allow developers to public application in an app store, on top of governmental datasets.

[17]http://soheil4tuwien.github.io/GoDaaS/index.html

So far, we modeled seven possible business opportunities on government open data. In order to take such opportunities, governments should seek to ease any friction that limits developers' ability to build their tailor-made applications on top of these business models. We have proposed that governments provide their underlying data resources as Data Compute Units (DCUs) for the sake of seamless development. Civic developers compose and program those DCUs and expose them to services delivered in mobile Apps to citizens. We envision $Gov.\ DataComputeUnit$ to gain acceptance as a *de facto* standard of an open government data resource unit norm, through broad adoption. In this context, we extended our contribution to an open government data service model entitled *GoDaaS* as a government data processing platform, where civic programmers are motivated to develop applications for citizens. As an outlook, we plan to elaborate and extend the GoDaaS to build a development blueprints tailored for a specific application that logically combine, orchestrate, and consume government data services.

References

[1] M. Burgess, *In Search of Certainty - The Science of Our Information Infrastructure*, 1st ed. XtAxis Press, 2013.

[2] E. Commission, "Study on business models for linked open government data - BM4LOGD," https://joinup.ec.europa.eu/node/72473, 12 November 2013, [Online; accessed 24 June 2014].

[3] "Osterwalder, alexander; pigneur, yves; and tucci, christopher l. (2005) "clarifying business models: Origins, present, and future of the concept," communications of the association for information systems: Vol. 16, article 1."

[4] N. Shadbolt and K. O'Hara, "Linked data in government," *Internet Computing, IEEE*, vol. 17, no. 4, pp. 72–77, July 2013.

[5] M. Vafopoulos and M. Meimaris, "Weaving the economic linked open data," in *Semantic and Social Media Adaptation and Personalization (SMAP), 2012 Seventh International Workshop on*, Dec 2012, pp. 92–97.

[6] J. Hoxha and A. Brahaj, "Open government data on the web: A semantic approach," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2011 International Conference on*, Sept 2011, pp. 107–113.

[7] E. Kalampokis, E. Tambouris, and K. Tarabanis, "On publishing linked open government data," in *Proceedings of the 17th Panhellenic Conference on Informatics*, ser. PCI '13. New York, NY, USA: ACM, 2013, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/2491845.2491869

[8] M. Vafopoulos, "A framework for linked data business models," in *Informatics (PCI), 2011 15th Panhellenic Conference on*, Sept 2011, pp. 95–99.

[9] D. DiFranzo, A. Graves, J. Erickson, L. Ding, J. Michaelis, T. Lebo, E. Patton, G. Williams, X. Li, J. Zheng, J. Flores, D. McGuinness, and J. Hendler, "The web is my back-end: Creating mashups with linked open government data," in *Linking Government Data*, D. Wood, Ed. Springer New York, 2011, pp. 205–219. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-1767-5_10

[10] F. Leymann, "Linked compute units and linked experiments: Using topology and orchestration technology for flexible support of scientific applications," in *Software Service and Application Engineering*, ser. Lecture Notes in Computer Science, M. Heisel, Ed. Springer Berlin Heidelberg, 2012, vol. 7365, pp. 71–80. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30835-2_6

[11] B. Nicolae, P. Riteau, and K. Keahey, "Bursting the cloud data bubble: Towards transparent storage elasticity in iaas clouds," in *28th IEEE International Parallel & Distributed Processing Symposium*, Phoenix, AZ, 2014.

[12] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," *IEEE Transactions on Software Engineering*, vol. 24, no. 5, pp. 342–361, 1998.