An OSGi-based Model-driven Data Management Module for Robust Open Data Harvesting

Arun Prakash, Santosh Kumar Rajaguru, Nikolay Tcholtchev
Fraunhofer Institute for Open Communication Systems FOKUS, Berlin
{arun.prakash | santosh.kumar.rajaguru | nikolay.tcholtchev}@fokus.fraunhofer.de

Abstract: The German governmental open data platform, GovData.DE, launched in February 2013, has served as the single point to access open governmental data. The outcome of this was a single metadata format, called the Open Governmental Metadata (OGD) that servers as the common standardized metadata structure for open government data in German speaking countries. The simplicity and extensibility of OGD allows it to be used for sharing open data from other domains, such as the cultural or educational domains. However, this requires that big data facets be addressed by the current Open Data Platform (ODP) implementation. Presently, CKAN is the de facto system for open data harvesting, storage and management. The open data platform in its current form is built on top of CKAN. The current implementation of harvesters and data management operations are highly CKAN dependent, which is a problem. Thus, in this paper, we present a novel ODP architecture based on OSGi and model-driven engineering that provides decoupling between the data management and CKAN. We showcase how our architecture addresses the required scalability and modularity concerns and provides a pathway to address the big data requirements.

Keywords: Open Data, Data Harvesting, OSGi, MDE, big data, CKAN, ODP, OGD

1. Introduction

The Open Data Strategy of the German Government is supported by the German governmental data portal (GovData.DE) [1], which was launched by the German *Federal Ministry of the Interior* in February 2013. The design, development and deployment of the portal were carried out by Fraunhofer Institute for Open Communication Systems FOKUS [2]. This study of Open Data solely depends upon the data providers as well as the interfaces for harvesting the data. To achieve this functionality, the harvesting mechanism was developed with the help of python environment.

Figure 1 illustrates the ODP architecture of GovtData.de. As, CKAN (Comprehensive Knowledge Archive Network) [3] is the de-facto standard for managing Open Data sets, it is used as the backend for harvesting the data. CKAN, which is developed by the Open Knowledge Foundation [4], is also being used by a set of Open Data portals across the globe, such as the *Open Data Portal of UK* [5] and the *Open Data Portal of Berlin* [6]. It was also used as a data hub in various research projects such as the EU-FP7 Open Cities project [7] or the Fraunhofer internal GeMo project [8] on collaborative electric mobility in Smart Cities. On top of CKAN, a Liferay [9] portlet container [10] was used to develop some components which enable the user friendly interactions between the community (data providers, app developers, data journalists etc.) and the CKAN registry in the backend.

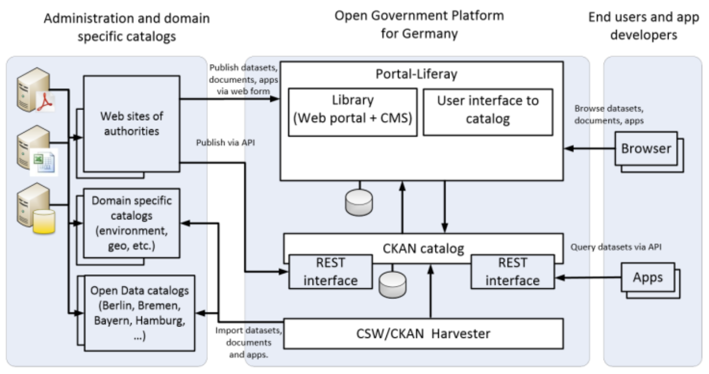
Administration and domain specific catalogs
Open Government Platform for Germany
End users and app developers
Web sites of authorities
Domain specific catalogs (environment, geo, etc.)
Open Data catalogs (Berlin, Bremen, Bayern, Hamburg, ...)
Publish datasets, documents, apps via web form
Publish via API
Import datasets, documents and apps.
Portal-Liferay
Library (Web portal + CMS)
User interface to catalog
CKAN catalog
REST interface
REST interface
CSW/CKAN Harvester
Browse datasets, documents, apps
Browser
Query datasets via API
Apps

Figure 1: Overall architecture of the open-data platform established for GovData.DE [11]

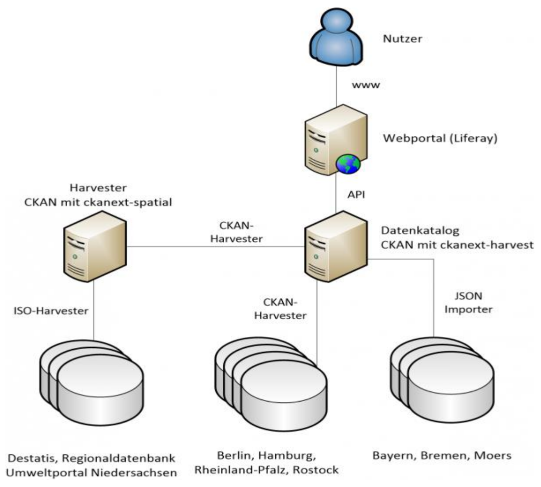
Nutzer
www
Webportal (Liferay)
API
Harvester CKAN mit ckanext-spatial
CKAN-Harvester
Datenkatalog CKAN mit ckanext-harvest
ISO-Harvester
CKAN-Harvester
JSON Importer
Destatis, Regionaldatenbank Umweltportal Niedersachsen
Berlin, Hamburg, Rheinland-Pfalz, Rostock
Bayern, Bremen, Moers

Figure 2: Data Management in GovData.DE [11]

In this paper we present a new data management architecture that has been upgraded from python /CKAN extension to Java OSGi framework to avoid scalability and modularity issues.

The rest of this paper is organized as follows: Section 2 briefs on our experiences in data harvesting in the context cultural data using OSGi framework and how the OSGi solves the issues related to modularity, scalability unlike the CKAN extensions. Section 3 elucidates on the need for a Model-Driven Engineering (MDE) approach to the development of the harvesters and presents the potential benefits of using MDE in the ODP. In Section 4 we present our novel architecture that incorporates modularity and scalability aspects. Finally in Section 5, we present our conclusions and insights in to the future steps that we intend to take.

2. Need for an OSGi-based Data Management Module

The current architecture employed for harvesting open-data, as applied in GovData.DE case, is illustrated in Figure 2. As it can be observed from the figure, the architecture employs the use of CKAN [3] for harvesting and managing open data provided by the various data providers. While this current approach has been successful and efficient, the question arises as to need to change this approach and upgrade it with the proposed Open Service Gateway Initiative (OSGi) [12] framework based data management approach, described in Section 4. In this section, we discuss this need to migrate from a python-based (CKAN) data management module to a more robust OSGi-based model-driven data management module.

One of the key indicators of a successful software platform lies in its ability to adapt to new requirements and to an ever changing technological landscape. Thus, a software platform/framework needs to be adaptive, scalable, modular in nature and configurable for it to be sustainable and successful. As the growth and adoption of open-data is projected to be exponential, the issue of *big data* [13] comes into the picture while harvesting and managing open data. For instance, at the moment, the number of *datasets* harvested in the GovData.DE [1] platform and Data.Gov.UK [5] is less than ten thousand and twenty thousand records respectively. However, at the same time, the number of datasets hosted by Europeana [14] is 33 million. This is a significant number and thus *big data* [13] concepts need to be employed to harvest and manage datasets from such providers. Thus the need to employ software that is reliable, scalable and allows for distributed processing of large datasets is evident. At the moment, the management of open data is tightly coupled to CKAN. The status of CKAN as the de facto standard for harvesting data management may change in the future and thus there would be a huge overhead to migrate harvesters written in python as CKAN extensions to a new technology. The alternative is to use an OSGi-based modular mechanism whereby the harvesters would be developed as modules that can be updated independent of the backend and data providers. The benefits of using OSGi are well known and the relevant aspects to data harvesting are described below.

As described in [15], *OSGi framework provides a modular system as well as a service platform which helps to implement a complete and dynamic model by reducing complexity for large scale applications*. The framework creates pluggable components that are not tightly coupled to each other but are rather cohesive in nature. Thus the issues commonly encountered in java application development, such as the JAR Hell [16], class path loading problems, dependency management, etc. are nicely dealt within OSGi. Furthermore, it supports *Hot Deployment* which eases the extension of the application. Developing with OSGi creates Bundles [17], which are called as modules, which interact with each other through well-defined services. OSGi provides a mechanism for dynamically managing the life cycle events such as activating, stopping and installing the bundles without shutting down the complete system. This solves the write once, run everywhere issue by using only those classes available on all environments and by activating the bundle only when code is available in the execution environments. This is an essential property for big data systems and is unfortunately not available in CKAN.

Thus, in order to achieve optimal results in terms of performance, scalability as well as adaptability the harvesting platform developed with OSGi framework is more suitable than a CKAN based system.

3. Need for a Model-driven Data Management Module

While OSGi provides the required modularity and scalability for the data management platform, the issue of semantics is not addressed. The CKAN based architecture, illustrated Figure 1 and Figure 2, do not have the notion of models, ergo data semantics and knowledge. The knowledge is somehow left back with the data provider during the harvesting of the data. While the OGD data schema [18] provides some kind a meta-model, this is not sufficient. The CKAN schema/OGD meta-data schema can only provide syntactical checks and not semantic validation. This is a major drawback as the data is only as good as its meta-model.

With Model-driven Engineering (MDE), some of the facets that need to be explicitly implemented or were unavailable are readily available. With MDE, the data furnished by the data providers and the data stored in the backend (CKAN or some other system) is expressed as a model based on a particular meta-model, as depicted in Figure 3. Thus facets such as model validation that had to explicitly developed are now inherently available due to nature of the model and meta-model relationship. Thus MDE leads to a meaningful validation of the datasets [20].

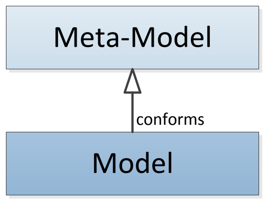

Figure 3: MDE - Meta-Model to Model Relationship

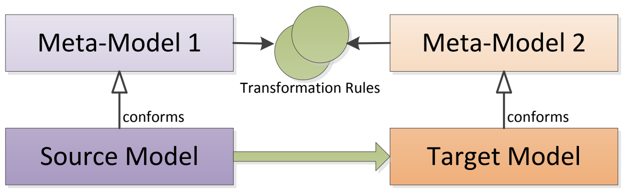

Figure 4: MDE - Model transformation from source to target

Furthermore, with MDE, data provided by one data provider (source) can be easily transformed to another (target) through the specification of transformation rules defined at the meta-model level. Thus, any change in the meta-model immediately reflects in the transformation rule. Thus MDE is less error prone than other mechanisms [20]. Usage of frameworks such as the Eclipse Modeling Framework (EMF) [19] simples MDE as it automatically generates the java code for development. Thus the developer of the data management module can focus on the more difficult issues related with big data than the actual data itself [20]. An exhaustive list of all the benefits of using model-driven development is described in [20].

Thus due to these reasons, the current nature and architecture of the data management module is not sustainable and needs to be incorporate OSGi and MDE concepts. In the section below, we describe our new OSGi-based model-driven architecture for data management in the open data platform. The architecture has been successfully tested in the context of harvesting cultural [25] and educational metadata [24].

4. OSGi-based Model-driven Data Management Module

Figure 5 illustrates the new harvesting and data management mechanism based on Java OSGi architecture. For each data provider, a single harvester module that extends the abstract super *harvester* is implemented. It is the concrete harvester that performs the individual harvesting of datasets from data sources. As illustrated in Figure 5, the data management mechanism is highly modularized. The overall workflow is depicted in Figure 6. As pictured, harvesting data from a source involves six simple steps. First the data is retrieved from the data provided. It is immediately validated for semantic and syntactical errors. If there are any errors, they are recorded and automatically reported back to data provider. Depending on the severity of the error and the configuration of the harvester, further actions are carried out. The input data is then transformed to the OGD meta-data schema. The transformed datasets are validated once again to check for transformation errors and finally pushed to the backend, which is CKAN in this case. All the modules are based on the OSGi extension mechanism and are managed by the Harvester Controller module (HCM). The HCM manages the configuration, scheduling and operation of the individual harvesters. The HCM is responsible for the instantiation and control of the individual harvesters. HCM instantiates the corresponding harvesters with the help of extension mechanism which incorporates the *Dependency Inversion Principle* [21] through the abstract *harvester* class. All modules, namely the *DataProvider*, *Validator, Transformer, BackendProvider and Harvester* are designed based on the based on SOLID design principles [21]. Thus it can accommodate various types and this increases the cohesiveness of the application. Finally, *Authentication &*

Authorization mechanisms are inbuilt to safeguard the security and integrity of the harvesters. All modules inherently work on models instead of raw data. ModelBus [22] serves as the model integration framework and provides the internal storage of the harvester related operations.

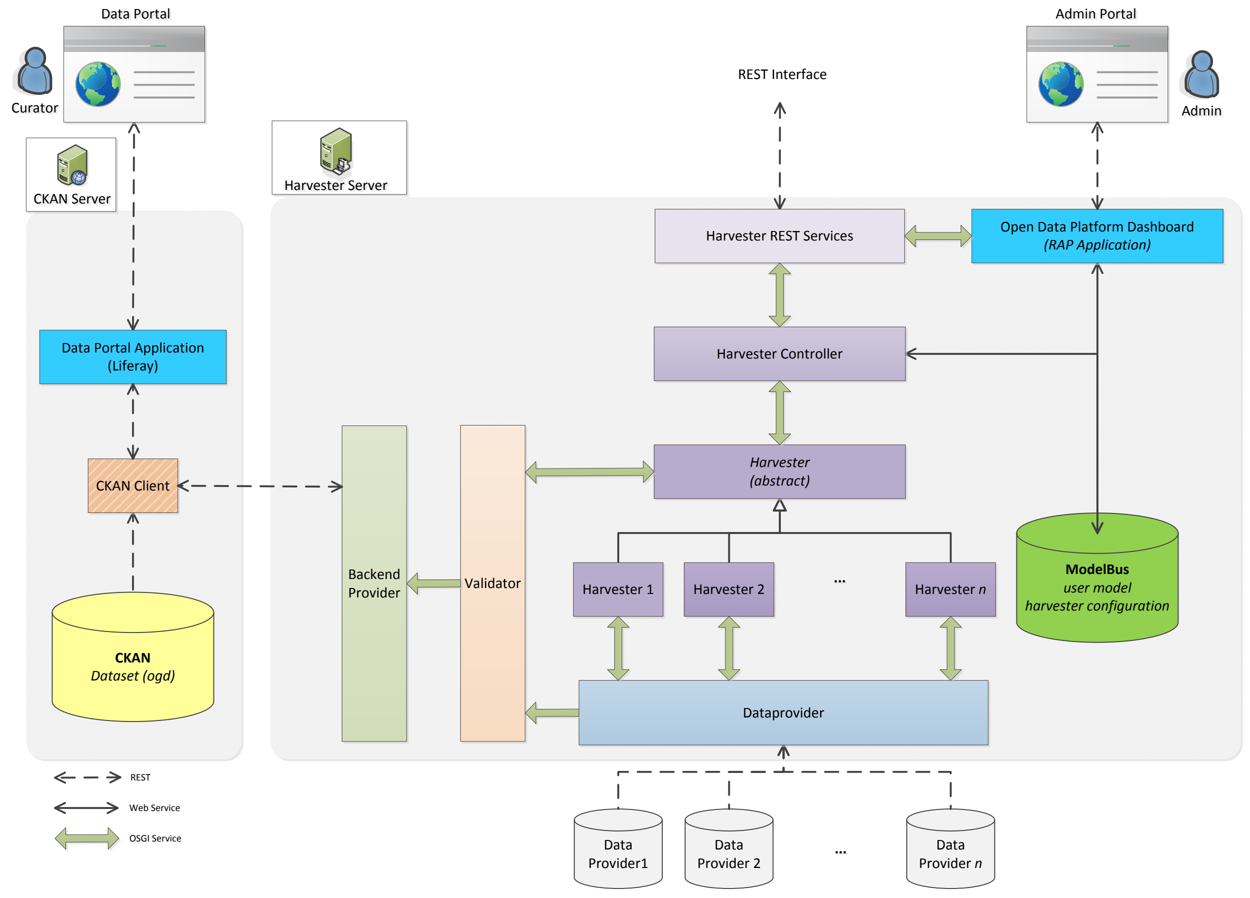

Figure 5: OSGi-based Model-driven Data Management Module

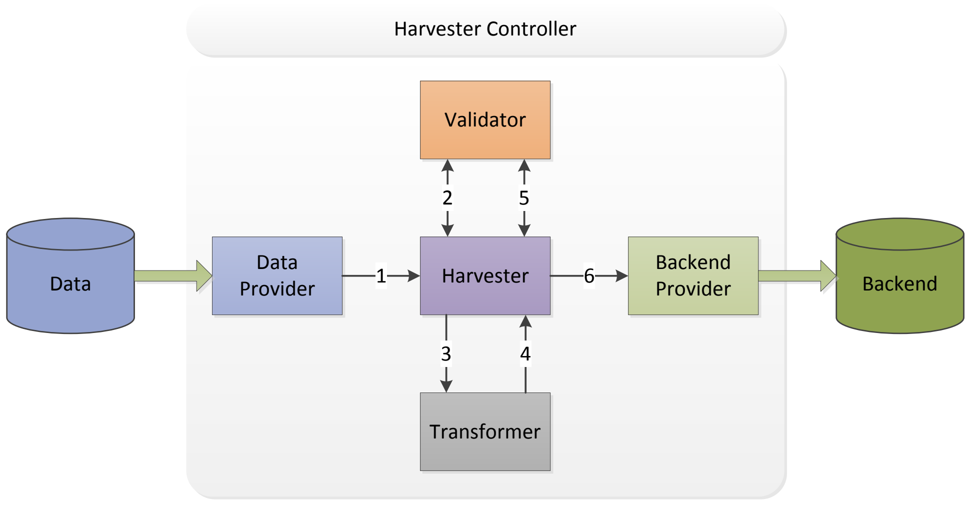

Figure 6: Modular workflow of the Harvester Controller

Finally, along with the OSGi console, the harvester application uses the Remote Application Platform (RAP) [**Error! Reference source not found.**] application to provide a thin client with a rich widget set to administer the harvesters.

5. Conclusions

In this paper, a new OSGi-based model-driven architecture to harvest and manage open data was presented. While the work carried out in previous open data projects is significant, the metadata harvesting requirements have shifted in the direction of big data. Thus, there is a need to handle and develop harvesters in a new way that takes this shift into consideration. OSGi and MDE provide promising solutions by moving the harvester logic embedded as python based CKAN extensions out of CKAN to a modularized system that can be independently developed irrespective of the changes in CKAN or introduction of a new backend for managing open data. In conclusion, the usage of OSGi and MDE would lead to the scalable and robust open data platform that is ready for commercial exploitation. The architecture was validated by harvesting open cultural and education metadata. In the future, we intend to experiment with big data systems such as Hadoop [23] in parallel with CKAN, as alternatives to store and manage open data.

6. Acknowledgement

This paper is partially based on the work being carried out in the EAGLE [24] and TAG CLOUD [25] projects.

7. References

[1] (2013, Feb) GovData.DE. Online. Federal Ministry of the Interior. [Online]. Available: www.govdate.de

[2] Fraunhofer FOKUS. Online. [Online]. Available: www.fokus.fraunhofer.de

[3] CKAN. Comprehensive Knowledge Archive Network. [Online]. Available: www.ckan.org

[4] Open Knowledge Foundation. Online. [Online]. Available: www.okfn.org

[5] Data.Gov.UK. Online. [Online]. Available: www.data.gov.uk

[6] Berlin Open Data Portal. Online. [Online]. Available: www.daten.berlin.de

[7] E. Lapi, N. Tcholtchev, L. Bassbouss, F. Marienfeld, and I. Schieferdecker, "Identification and Utilization of Components for a Linked Open Data Platform," in *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, July 2012, pp. 112–115.

[8] N. Tcholtchev, B. Dittwald, T. Scheel, B. Zilci, D. Schmidt, P. Lammel, J. Jacobsen, and I. Schieferdecker, "The Concept of a Mobility Data Cloud: Design, Implementation and Trials," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, July 2014, pp. 192–198.

[9] J. Yuan, *Liferay Portal Enterprise Intranets: A Practical Guide to Building a Complete Corporate Intranet with Liferay*, ser. IT Pro. Packt Pub., 2008. [Online]. Available: http://books.google.co.in/books?id=3nismAEACAAJ

[10] *JSR 268: Java Specification Requests 286*, Oracle Std. [Online]. Available: https://www.jcp.org/en/jsr/detail?id=286

[11] F. Marienfeld, I. Schieferdecker, E. Lapi, and N. Tcholtchev, "Metadata aggregation at govdata.de: An experience report," in *Proceedings of the 9th International Symposium on Open Collaboration*, ser. WikiSym '13. New York, NY, USA: ACM, 2013, pp. 21:1–21:5. [Online]. Available: http://doi.acm.org/10.1145/2491055.2491077

[12] Open Service Gateway Initiative. [Online]. Available: www.OSGi.org

[13] M. D. de Assunção, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big Data Computing and Clouds: Challenges, Solutions, and Future Directions," *CoRR*, vol. abs/1312.4722, 2013. [Online]. Available: http://arxiv.org/abs/1312.4722

[14] B. Valtysson, "Europeana : The digital construction of europe's collective memory," *Information, Communication & Society*, vol. 15, no. 2, pp. 151–170, 2012. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/-1369118X.2011.586433

[15] Wikipedia. OSGi. [Online]. Available: http://en.wikipedia.org/wiki/OSGi

[16] JAR Hell. [Online]. Available: http://en.wikipedia.org/wiki/Java_Classloader#JAR_hell
[17] OSGi Bundles. [Online]. Available: http://www.OSGi.org/javadoc/r4v43/core/org/OSGi/framework/Bundle.html
[18] OGDD meta-data scheme. [Online]. Available: https://github.com/fraunhoferfokus/ogd-metadata
[19] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.
[20] J. D. Haan. (2005, November) 15 reasons why you should start using Model Driven Development. Online. [Online]. Available: http://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/
[21] Wikipedia. SOLID (object-oriented design). [Online]. Available: http://en.wikipedia.org/wiki/SOLID_%28object-oriented_design%29
[22] A. Aldazabal, T. Baily, F. Nanclares, A. Sadovykh, C. Hein, M. Esser, and T. Ritter, "Automated Model Driven Development Processes," in *Proceedings of the ECMDA workshop on Model Driven Tool and Process Integration*. Fraunhofer IRB Verlag, Stuttgart, 2008, iSBN: 978-3-8167-7645-1.
[23] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.
[24] "EAGLE - Enhanced Goverment Learning," Online. [Online]. Available: http://www.eagle-learning.eu/
[25] "TAG CLOUD - Technologies lead to Adaptability & Lifelong Engagement with Culture Throughout the Cloud," Online. [Online]. Available: http://www.tagcloudproject.eu/