A Toolset for Supporting Evolution and Preservation of Linked Data: the DIACHRON approach

Panagiotis Hasapis, Danae Vergeti, Aggelos Liapis, Antonis Ramfos
{Panagiotis.Hasapis, Danae.Vergeti, Aggelos.Liapis, Antonis.Ramfos}@intrasoft-intl.com, INTRASOFT International, 2b Rue Nicolas Bove, Luxembourg, Luxembourg

Giorgos Flouris, Kostas Stefanidis, Ioannis Chrysakis, Yannis Roussakis
{ fgeo, kstef, hrysakis, rousakis }@ics.forth.gr, FORTH ICS, N. Plastira 100 Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

Marios Meimaris, George Papastefanatos, Yannis Stavrakas, Christos Pateritsas, Theodora Galani
{m.meimaris, gpapas, yannis , pater , theodora }@imis.athena-innovation.gr, Research Center Athena, Artemidos 6 & Epidavrou, Marousi 15125, Greece

Peter Buneman, James Cheney, Slawomir Staworko, Stratis D. Viglas
{opb, jcheney,sstawork, sviglas}@inf.ed.ac.uk, School Of Informatics, Informatics Forum, 10 Crichton Street EH8 9AB, Edinburgh, United Kingdom

Jeremy Debattista, Natalja Friesen
jeremy.debattista@iais-extern.fraunhofer.de, Enterprise Information Systems, University of Bonn, Institut für Informatik III Römerstraße 164 53117 Bonn, Germany

Loïc Petit
loic.petit@data-publica.com, Data-Publica, 22 rue Chauchat, 75009 Paris, France

Simon Jupp, Tony Burdett
{jupp, tburdett}@ebi.ac.uk, EMBL – European Bioinformatics Institute, Welcome Trust Genome Campus, Hinxton, UK.

Robert Isele, Knud Möller
{robert.isele, knud.moeller}@eccenca.com, eccenca GmbH, Neumarkt 20, 04109 Leipzig, Germany

Abstract.

Over the course of the last few years, there has been a vast and rapidly increasing quantity of scientific, corporate, government and crowd-sourced data, published on the emerging Data Web that has been created for open access. Open Data is expected to play a catalyst role in the way structured information is exploited in the large scale. This offers a great potential for building innovative products and services that create new value from already collected data. Open data published according to the Linked Data Paradigm is essentially transforming the Web from a document publishing-only environment, into a knowledge ecosystem where users have become active data aggregators and generators themselves. A traditional view of digitally preserving them by pickling them and locking them away for future use, like groceries, would conflict with their evolution. There are a number of approaches and frameworks, such as the LOD2 stack, that manage a full life-cycle of the Data Web. More specifically, these techniques are expected to tackle major issues such as the **synchronisation problem** (how can we monitor changes), the **curation problem** (how can data imperfections be repaired), the **appraisal problem** (how can we assess the quality of a dataset), the **citation problem** (how can we cite a particular version of a linked dataset), the **archiving problem** (how can we retrieve the most recent or a particular version of a dataset), and the sustainability problem (how can we spread preservation ensuring long-term access).

In this paper we describe DIACHRON, a unified semantic platform for supporting the evolution and preservation of Linked Dataset. We describe modules that tackle the previously mentioned issues. With regard to the synchronization problem, our approach allows the identification and analysis of the evolution of a dataset, in an efficient, user-friendly and customizable manner. The proposed solution allows the execution of queries spanning multiple versions, as well as queries related to the evolution itself (rather than just the data). For the citation problem, we describe a rule-based mechanism for specifying, extracting, and assigning citable persistent identifiers to diachronic resources. A sequential process is implemented to efficiently assess a dataset's quality, providing the user with the necessary quality metadata and quality problem report as a bonus, in order to keep track of the appraisal problem. With regard to the archiving problem, we have designed and developed a conceptual model that captures both structural and semantic aspects of evolving data, thus enabling evolution management at different granularity levels. Based on this, we have implemented a query language as an extension of SPARQL that inherently tackles querying evolving entities and their changes across time. Supporting this platform we provide three real-life use-cases; a business use-case, a life science use-case, and an Open Data use-case.

Keywords: linked data, data evolution, digital preservation, appraisal, provenance problem, data curation

1. Introduction

Over the course of the last few years, there has been a vast and rapidly increasing quantity of scientific, corporate, government and crowd-sourced data, published on the emerging Data Web that has been created for open access. The basic advantage oft he Data Web is that facts are recorded rather than documents, which become the basis for the discovery of new knowledge that is not contained in an individual source. In particular, open data published according to the Linked Data Paradigm are essentially transforming the Web from a document publishing-only environment, into a knowledge ecosystem where users have become active data aggregators and generators themselves.

Although Linked Open Data (LOD) provide a unique and flexible paradigm for publishing data, preserving this type of knowledge and information poses several challenges not actually addressed by past and ongoing Linked Data and Digital Preservation projects:

- LOD are Structured: Unlike documents, we need to manage not individual facts but entire LOD datasets representing real-world entities for which additionally constraints (e.g., name uniqueness) may hold. Moreover, LOD may be interconnected through typed links when they refer to the same or related real-world entities. This calls for effective entity recognition and co-reference methods to rank LOD datasets according to their quality for guiding crawling and appraisal. It also stresses the need for preserving an entire network of interconnected LOD datasets which may prove to be useful for future analyses.

- LOD are Dynamic: Unlike closed settings in which data changes are communicated via notification mechanisms, LOD evolution in the Data Web can be periodically observed through crawling or by other mechanisms such as Publish/Subscribe. In addition, understanding the changes of evolving LOD datasets and repairing their potential inconsistencies requires methods for recognizing complex change patterns beyond low-level atomic changes. In particular, discovering LOD differences (deltas) and representing them as first class citizens with structural, semantic, temporal and provenance information is vital in various tasks such as the synchronization of autonomously developed LOD versions, or visualizing the evolution history of a particular LOD dataset. For example, biomedical data analyses and data releases consuming LOD are version/date stamped and cannot be repeated or fully understood unless the correct versions of supporting data are available.

- LOD are Distributed: By definition LOD production, processing and consumption are activities distributed among several actors worldwide. Today we have reasonable methods for protecting our data from physical destruction, but this is no guarantee against the economic collapse of the organization that maintains the data.

In this respect, the authors' consider addressing the following challenges arising when preserving Linked Open Data (LOD) of varying quality actually published on the Data Web:

- How can one monitor changes of third-party LOD datasets released in the past (the evolution tracking problem) or how can newly released versions be considered by ongoing data analysis processes (the change synchronization problem)?
- How can we understand the evolution of LOD datasets with respect to the real world entities they describe (the provenance problem) and how can various data imperfections (e.g., granularity inconsistencies) be repaired (the curation problem)?
- How we can assess the quality (temporal and spatial) of harvested LOD datasets in order to be able to decide which and how many versions of them deserve to be further preserved (the appraisal problem)?
- How do we cite particular versions of a LOD dataset (the citation problem), and how will we be able to retrieve them when looking up a reference in the form in which we saw it – not the most recently available version (the archiving problem)?
- How can we spread preservation costs to ensure long-term access even when the original motivation for publishing has changed (the sustainability problem)?

In this paper, the authors present DIACHRON, a framework and set of services that will allow data stewards, curators (and people working in the data publishing domain) to monitor changes, to sustain and manage versions of different datasets. All separate components that help resolve each particular aforementioned problem can be used autonomously or in form of an integrated architecture as web services in order to facilitate the ease of deployment and usage. First, a short description of three use cases in relation to our paradigm will be presented. Secondly, the overall architecture will be presented, followed by a set of core services described. In the end, a set of results of the usage of those services will be provided.

2. Use Case Description

2.1. Business Scenario Case

In this scenario, a large manufacturer in the automotive industry is developing a web-based customer portal that will allow users to browse their range of current and historical models, read related news, browse the company's history, find like-minded people through car clubs, etc.

A range of data sources from different divisions of the company have been identified as relevant for implementing the portal: an extensive car model database with detailed technical features for each model, a technical term database, various news channels, club listings and history data. In addition, several external LOD and non-LOD data sources have been identified which will provide further context. Examples are DBpedia, Linked Geodata and car review sites.

Since preparing the data for the portal is largely a data integration problem, and since several external data sources already exist as LOD, Linked Data has been chosen as the development approach for the data backend. To this end, an infrastructure has been set up which converts all internal and external non-LOD data to LOD and then establishes links between the datasets.

Because cars are the main interest of the expected portal users, the car model database will act as the datahub in the application's LOD: most other datasets will link their resources to the relevant car models (news items about those models, clubs interested in a particular model, etc.). Tracking the evolution of the car model dataset is therefore crucial to ensure that the references from all other datasets stay intact (*evolution and change synchronization*). Additionally, the external LOD data sources need to be *appraised*, before they can be included in the production code of the portal.

2.2. Life Sciences Case

The life sciences have been quick to adopt LOD as a mechanism for publishing biological data on the Web. This data is exploited by industry and academia to advance out understanding of biology and used in the development of new drugs and areas such as translational medicine. Organisations like the EMBL European Bioinformatics Institute are dedicated to the preservation and curation of this data and serve it back to the community through a wide range of services. EMBL-EBI has already begun to embraced LOD technologies through the release of their LOD RDF platform [Jupp 2013] and invest heavily in the annotation and curation of data to add value through the use of ontologies [Malone 2010].

This large-scale annotation presents a challenge especially when you consider how the data and the ontologies continue to evolve as the technologies used to generate the data change and our understanding of the underlying biology progresses. The ability to track changes in the data helps to reduce the overall cost of curation as errors are can be detected more readily and repaired appropriately. Having a robust strategy for handling data provenance also improves overall data transparency when new hypothesis are generated from large data analysis and integration.

The Samples Phenotypes and Ontologies Team (SPOT) at EMBL-EBI are currently exploiting the Diachron platform to track changes in the ontologies and ontology-data mappings. By using the Diachron platform, SPOT are able to develop novel applications that assist the database curators in dealing with changes in the ontologies that impact on downstream annotations in the data.

2.3. Open Data Case

Using open data, it is possible to create a dashboard to expose the situation of a specific region in terms of economy or demography. Sources for this dashboard uses public datasets from the governments and / or statistical institutions like Eurostat for Europe or more local ones like INSEE (France) or StatBel (Belgium).

Unfortunately, most of the data is not published as LOD but by raw files like CSV, XLS, XML, SDMX etc… Although, this data is mostly multidimensional and Open Data platforms such as Data-Publica or DataMarket aggregate those and structure them in a unified way. Therefore, it is possible to use the RDF Data Cube Vocabulary [http://www.w3.org/TR/vocab-data-cube/] to model the data as structured triples that references common concepts from the LOD, say cities or countries. Making our approach viable to those partners.

Data preservation is key in this domain because the concepts data sets refer to evolve over time. For instance, a city can be created, renamed or merged over time; countries are split or reunified; regions can also be redefined from scratch. Hence, if we want to track the demography of a country over a long period of time like in this use case, it is crucial to understand the evolution of the country over time. For instance, if we want to plot the demography of Germany from 1970 until now, we have to take into account the fact that Germany was in fact two countries until 1990. This platform will help to understand and keep track of those similar changes.

Finally, the upstream sources can evolve over time as well. Dimensions can be added to datasets (or removed), license terms or identifiers can change. Currently, curators must deal with these problems using ad-hoc routines.

3. Integrated Platform Overview

In DIACHRON, a three layered architecture will be implemented and deployed. Figure 1 presents a conceptual overview of the system. The two layers of the system (which of our pilot applications will adjust to their existing systems) are the ***Platform Layer*** (Service Inventory) and the ***Integration Layer***.

Platform Layer - Service Inventory. The first (bottom) layer is the core DIACHRON platform and the SOA perspective acts as a service inventory. This layer includes the core functionality of the platform that corresponds to the research challenges of the project. The services of this layer are independent of each other, namely none of them uses another service of this layer to implement its functionality. There isn't any kind of direct data or message exchange between them. This allows for the system to operate even in the case where some services are disabled. Of course in such a case the system will not be able to provide the overall functionality to its full extent, but whatever adjustments need to be made are limited to the Service layer (integration). The independence of this layer according to the service orientation paradigm is pursued in order to benefit from the advantages of service orientation. More specifically, it provides a larger degree of flexibility in the technologies selected for each service module and its associated deployment options.

Integration Layer - Service Composition. The second (middle) layer is a service composition and integration layer and acts as a controller/orchestrator of the functionality of the core layer services in order to provide more complex services to the users. This layer is a middleware layer with a dual role. It encapsulates the services of the core layer and re-exposes the DIACHRON functionality by composing more complex services and by homogenizing the service contracts and endpoints. Additionally, it provides common functionality such as security and auditing control, logging, storage area for configuration information, etc. The components of this layer act as orchestrators of the core functionality provided by the platform layer. Their role is to execute specific workflows that usually involve more than one core service from the platform layer in order to create service compositions and expose more composite functionality to the pilot applications that use them.
In the next sections we will present the web service and components that belong to each of the aforementioned layers of the system.

4. Services Description

4.1. Integration Layer

4.1.1. Query API

This component exposes an API and acts as the primary data endpoint of the platform. It performs queries to the archive using the DIACHRON query language and also uses the citation service in order to dereference and return datasets to the users.

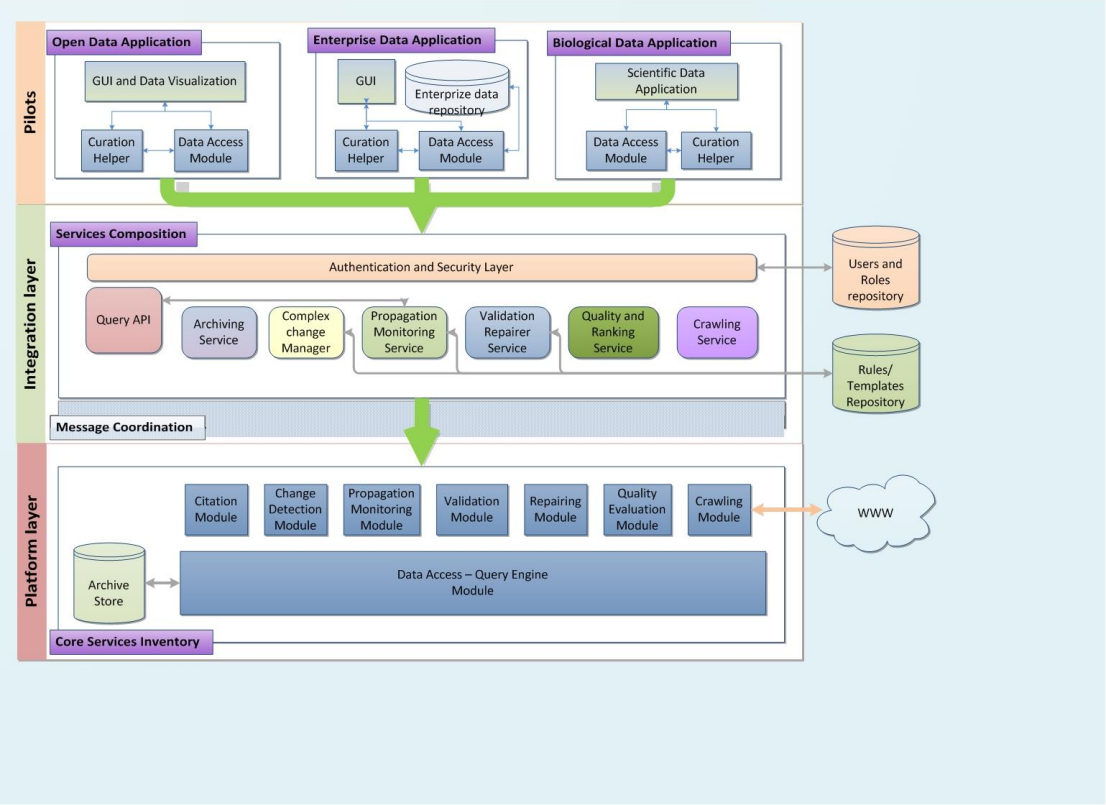

Figure 1 - Integrated Platform Architecture

4.1.2. Archiving Service

This component is the primary data input channel, as datasets or temporal versions of them that need to be stored in the archive must go through this component. As a service composition it uses the change detection service to detect changes and feed them to the Data storage module in case of an already existing dataset.

4.1.3. Quality Service

This component exposes the functionality of the quality evaluation module and also provides datasets ranking functionality based on the quality evaluations of a group of datasets or the series of temporal versions of the same dataset.

4.1.4. Authentication and Security Mechanism

This component provides authentication functionality for the system and also, if needed, more complex security and auditing services (selective access to dataset, logging, etc.).

4.1.5. Repairing Service

This component is responsible for validating and, if needed, repairing datasets according to sets of validation and repairing rules.

4.1.6 Cleaning Service

The goal of the cleaning component is detection of incomplete, incorrect, inaccurate facts in the data and then replacing, modifying, or deleting them in order to improve data set quality. The cleaning service is designed in a way to provide as much automated operation as possible. For that the identification of quality problems and generation of cleaning suggestions are done automatically in the background. The concrete user advices and data transformation rules are maintained in an extensible ontology.

4.2. Core Platform Layer

4.2.1. Change Detection Module

The change detection module is at the core of our approach, and its role is to capture the dynamics of LOD datasets by identifying the changes occurring

among any pair of dataset versions. There are two main components of the change detection module.

The first component is the list of supported changes, which consists of two levels. The lower level, called *simple changes*, contains fine-grained changes which are adequate for the data model at hand (i.e., they are data-model-specific); the definition of simple changes provably guarantees that detection is deterministic and complete in any possible evolution scenario. The upper level, called *complex changes*, contains application-specific, customized changes, suitable for capturing changes that are important, frequently-occurring or abnormal for the specific context of usage; defining complex changes is done by the user at run time, through an adequate interface.

The second component of the change detection module is the *ontology of changes*, which is a specially designed ontology for storing the changes that have been detected among versions. Storing detected changes as instances of the ontology of changes allows the use of linked data principles to easily navigate between the changes, the different versions, and the data itself, thereby supporting longitudinal queries, as well as queries involving both changes and data. This way, detected changes are treated as first-class citizens in the diachronic repository. For example, consider a query requesting "all countries for which the unemployment rate of their capital city increased at a rate higher than the average increase of the country as a whole, and the corresponding period in which this happened". This query requires access to the data itself (for identifying the countries and their corresponding capital cities) and to the evolution history (through which one can determine the rate of increase of the unemployment rate in each country and city); in addition, the last clause of the query requires returning all periods (i.e., pairs of versions) in which this happened, therefore the query is longitudinal (spanning over all available versions rather than over a specific pair of versions).
The change detection module was implemented in a robust and flexible manner that allows easy customization. This customization may be necessary in cases where a new data model (or a new set of simple changes) needs to be supported; that would require some adaptation of the configuration files, but no changes in the code itself. More details on this work, including the associated theoretical framework and an experimental evaluation can be found at [rousakis]; this work can be seen as an extension and generalization of [papavassiliou], with significantly improved computational properties.

4.2.2. Quality Evaluation Module

The Quality Evaluation Module (QEM) is responsible for assessing the quality of linked datasets available in the archive. This backbone of this module is *Luzzu*[1], a quality assessment framework for linked data [luzzu]. Luzzu provides an integrated platform that: (1) assesses Linked Data quality using a library of generic and user-provided domain specific quality metrics in a scalable manner; (2) provides query-able quality metadata on the assessed datasets; (3) assembles detailed quality reports on assessed datasets. The Luzzu infrastructure also enables the definition of domain-specific metrics, employing a semantic backend of ontologies to represent quality metadata about the assessed datasets in a standardised manner.

When QEM is invoked with one or more Diachronic datasets and metrics, the first process is to de-reify the dataset into a flat RDF representation in order to make use of Luzzu's efficient stream processing. Zaveri et al. [zaveri] described various metrics that are relevant to DIACHRON. When triples are streamed, metrics are executed in parallel in order to compute a quality result and to identify possible problems in the dataset that require cleaning or repairing. We are currently investigating probabilistic approximation techniques in order to improve the runtime of this module. Once the assessment is over, the module communicates the *quality metadata* and a *quality report* back to the integration layer for further assessment by the user.

4.2.3. Data Access Module

The archive module is the primary data channel, as datasets and their temporal versions that need to be stored in the archive must go through this component. As a service composition it creates and manages diachronic datasets and uses the change detection service to detect changes and feed them to the Data storage module in case of already existing datasets. The module's functionality is exposed via the HTTP protocol as the primary access mechanism of the archive through a RESTful web service API. The basic components the module employs are the Data Access Manager, the Store Connector, the Data Modeler, the Archive Optimizer and the Query Processor.

Incoming datasets conform to the DIACHRON dataset model [meimaris], while the Data Modeler component handles the dataset input functionality and data transformations from the DIACHRON dataset model to the native data model of the store and

[1] http://eis-bonn.github.io/Luzzu

vice versa, and consists of the Data Translator and the Data Loader. The Archive optimizer component enables optimization of the storage method of incoming datasets based on various archive strategies as shown in [4.1.2] and further discussed in [stefanidis] that are out of the scope of this paper. It performs analysis of the dataset characteristics and chooses the most efficient storage strategy based on metrics. The Query Processor component is the base mechanism for query processing and thus data access. Its query engine handles SPARQL queries, as well as queries formulated using the DIACHRON Query Language, which is a specialized extension of SPARQL designed to abstract the inherent structure and characteristics of the DIACHRON dataset model in order to make querying on DIACHRON datasets intuitive and uncomplicated.

Presently, the archive module is deployed on top of a Virtuoso 7.1 instance, but is implemented to be independent from the underlying storage technologies and can thus be used on top of a variety of stores from different vendors.

4.2.4. Citation Module

Citations are one of the most significant tools used in the creation and propagation of knowledge and therefore play a central role in DIACHRON as well. Through citations we can give attribution to the data point creator, convey information about the content, assign responsibility of ownership, and provide an identifier (either machine- or human-readable) for enhancing the longevity of data through persistent identification and retrieving the referenced work. DIACHRON incorporates a citation mechanism that not only satisfies these goals but also addresses the normal dataset provider requirements of being efficient and robust under change.

The citation mechanism is coupled with DIACHRON's data model. Every diachronic resource of the data model is inherently citable. Implementing citation consists of two substrates: (i) a description of the citation format, and (ii) a specification for identifying citable units in the diachronic dataset [buneman]. The citation format allows the system to generate the citation in whichever way the dataset user desires. The format contains a number of variables that are to be instantiated through evaluating path expressions in the hierarchical diachronic dataset. For instance, `{Citation[Author=$a]}` specifies a `Citation` with a `Author` attribute that is to be populated by binding variable `$a`. To valuate `$a` we need a path expression that is to be evaluated over a diachronic dataset and act as a generator. An example of such a path expression is `resource/book/author=$a` and may result in a citation like `{Citation[Author=Tesla]}` if `Tesla` is the ending point of the path expression.

Each generated citation is given a unique identifier that is guaranteed by DIACHRON to be stable for the lifetime of the datasets it cites. Additionally, the citation itself is stored along with its generating specification. This ensures that once a citation is looked up, DIACHRON can retrieve the associated information by "replaying" the citation generation mechanism.

5. DIACHRON's non-Functional Features

5.1.1. Performance – Response Time

The DIACHRON platform requirements regarding performance and response time may vary depending on the functionality. Archiving operations do not exhibit a great degree of need of high response time since these operations are mainly automated executed by system users of the system (pilot existing applications). On the other hand, the querying functionality will be used by human users and in some cases in order to provide results requested by web pages etc. In these cases the response time is critical.

5.1.2. Scalability

The amount of data especially in the Open Data and the Scientific Data use cases are expected to be significant. Given the high rates of increase of the available datasets, the platform utilised design approaches and technologies that are capable of scaling up to even larger data volumes.

5.1.3. Security-Privacy

The platform handles datasets that already public in the web, therefore privacy and security do not constitute a critical factor of the platform. The enterprise scenario is an exception as it includes sensitive corporate information. Consequently the platform contains a user management and access control functionality to prevent unauthorized access to such data.

5.1.4. Interoperability

The platform currently cooperates with the existing systems of each organization in order to provide the required functionality. Therefore, proper decisions have been taken during the modelling and design phase of the platform in order to ensure a high degree of interoperability. A loosely coupled, Ser-

vice-Oriented Architecture (SOA) has been adopted to achieve this.

5.1.5. Usability

Within DIACHRON usability issues are not ranked high because of the fact that the platform's user base does not include non-expert users. On the contrary pilot applications that visualize the datasets and their evolution will be used by web users. Consequently in the design of these applications, the necessary usability aspects have been tackled; especially in the evolution visualization features so as to seemly integrate with the rest of DIACHRON's GUI.

6. Conclusion and Future Work

The scope of this paper was to provide the description of operation and orchestration of the components that comprise the integrated DIACHRON platform. Its main purpose was to present the components, how they behave in an integrated environment, and how the communication will be conducted, by describing the messages exchanged.

The integrated DIACHRON Framework is based on Service Oriented Architecture (SOA) with the usage of message queues, which has been proven to be the most effective way to achieve synergy between a multitude of organisationally independent and technically heterogeneous services.

DIACHRON architectural design essentially relies on a shared semantic framework for capturing message meaning (data and functional semantics), as well as asynchronous messaging, to allow frictionless linking of heterogeneous systems.

The design and development decisions presented in this paper guided the implementation of the first prototype of the DIACHRON platform, which provided all the aforementioned functionality. During the second development cycle of the module, focus will be given to address performance and scalability issues that will arise from the use of the modules by the pilot applications as described in section 2. In the course of this second development cycle, certain design approaches might be revised and if so it will be reported in future publications.

Acknowledgement

This work has been co-funded by the DIACHRON project, a European Commission research program under Contract Number FP7-601043.

References

[buneman] Buneman, P. and Silvello, G. A Rule-Based Citation System for Structured and Evolving Datasets. IEEE Data Eng. Bull. 33(3): 33-41 (2010)

[jupp] Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., Gaulton, A., Gehant, S., Laibe, C., Redaschi, N., Wimalaratne, M. S., Maria Martin, Novère, L. N., Parkinson, H., Birney, E., and Jenkinson, A. M. The EBI RDF platform: linked open data for the life sciences. Bioinformatics (2014) 30 (9): 1338-1339 doi:10.1093/bioinformatics/btt765

[luzzu] Debattista, J., Londoño, S., Lange, C., & Auer, S. (2014). LUZZU-A Framework for Linked Data Quality Assessment. *arXiv Preprint arXiv:1412.3750*.

[malone] Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., and Parkinson, H. Modeling sample variables with an Experimental Factor Ontology.Bioinformatics (2010) 26 (8): 1112-1118 doi:10.1093/bioinformatics/btq099

[meimaris] M. Meimaris, G. Papastefanatos, C. Pateritsas, T. Galani, and Y. Stavrakas. Towards a Framework for Managing Evolving Information Resources on the Data Web. In PROFILES2014.

[papavassiliou] Papavassiliou, V., Flouris, G., Fundulaki, I. Kotzinos, D., Christophides, V. High-Level Change Detection in RDF(S) KBs. Transactions on Database Systems (TODS), 38(1), 2013.

[rousakis] Yannis Roussakis, Y., Chrysakis, I., Stefanidis, K., Flouris, G., Stavrakas, Y. A Flexible Framework for Defining, Representing

and Detecting Changes on the Data Web. ArXiv technical report, #1501.02652, 2015. Available at: http://arxiv.org/abs/1501.02652

[stefanidis] Stefanidis, Kostas, Ioannis Chrysakis, and Giorgos Flouris. "On Designing Archiving Policies for Evolving RDF Datasets on the Web." In Conceptual Modeling, pp. 43-56. Springer International Publishing, 2014.

[zaveri] Zaveri, A. et al. Quality Assessment Methodologies for Linked Open Data. In: Semantic Web Journal (2014). http://www. semantic-web- journal.net/content/quality- assessment-linked- data- survey. Forthcoming.