

6

Use Cases

**Kevin Feeney¹, Christian Dirschl², Andreas Koller³, James Welch⁴,
Dimitris Kontokostas⁵, Pieter Francois⁴, Sabina Łobocka⁶
and Piotr Bledzki⁶**

¹Trinity College Dublin, Ireland

²Wolters Kluwer Germany, Germany

³Semantic Web Company, Austria

⁴University of Oxford, UK

⁵University of Leipzig, Germany

⁶Wolters Kluwer Poland, Poland

6.1 Wolters Kluwer – Re-Engineering a Complex Relational Database Application

6.1.1 Introduction

The publishing industry is – like many other industries – undergoing major changes. These changes are mainly based on technical developments and related habits of information consumption.¹ The world of the customers has dramatically changed and as an information service provider, Wolters Kluwer wanted to meet these changes with the best solutions for the customers and their work environment.

Wolters Kluwer has already engaged for a couple of years in new solutions to meet these challenges and to improve all processes of generating good quality content in the backend on the one hand and to deliver information and software in the frontend that facilitates the customer's life on the other hand.

One of these frontend applications is a platform called JURION² – an innovative legal information platform developed by Wolters Kluwer Germany (WKD) that merges and interlinks over one million documents of content and

¹See e.g., this article about the information consumption in the US http://hmi.ucsd.edu/pdf/HMI_2009_ConsumerReport_Dec9_2009.pdf

²<https://www.jurion.de/de/home/guest>

data from diverse sources such as national and European legislation and court judgements, extensive internally authored content and local customer data, as well as social media and Web data (e.g., from DBpedia). In collecting and managing this data, all stages of the Data Life Cycle are present – extraction, storage, authoring, interlinking, enrichment, quality analysis, repair and publication. On top of this information processing pipeline, the JURION development teams add value through applications for personalisation, alerts, analysis, and semantic search.

The JURION use case is addressing both software life cycle and data life cycle. Therefore, their combination and integration is a key challenge within this use case. Still, currently both life cycles are highly independent from each other, which lead to a lot of errors and inefficient use of resources.

In order to address this challenge in a practical and pragmatic way, we have developed based on our daily operational experience two dedicated use case scenarios that shed a first light on the challenge and also on our view how to address it.

We have deliberately chosen one use case scenario that is triggered by the data life cycle and a second scenario triggered by the software life cycle. We also tried to describe common, yet not too complex situations, so that we could cover them in a sufficient granularity.

6.1.2 Problem Statement

JURION is an innovative legal information platform developed by Wolters Kluwer Germany that merges and interlinks over one million documents of content and data from diverse sources such as national and European legislation and court judgements, extensive internally authored content and local customer data, as well as social media and Web data (e.g., from DBpedia). In collecting and managing this data, all stages of the Data Life cycle are present – extraction, storage, authoring, interlinking, enrichment, quality analysis, repair and publication. On top of this information processing pipeline, the JURION development teams add value through applications for personalisation, alerts, analysis and semantic search. Based on the FP7 LOD2 project, parts of the Linked Data stack have been deployed in JURION to handle data complexity issues (see Figure 6.1). Currently, the software development process and data life cycle are highly independent from each other and require extensive manual management to coordinate their parallel development, leading to higher costs, quality issues and a slower time-to-market.

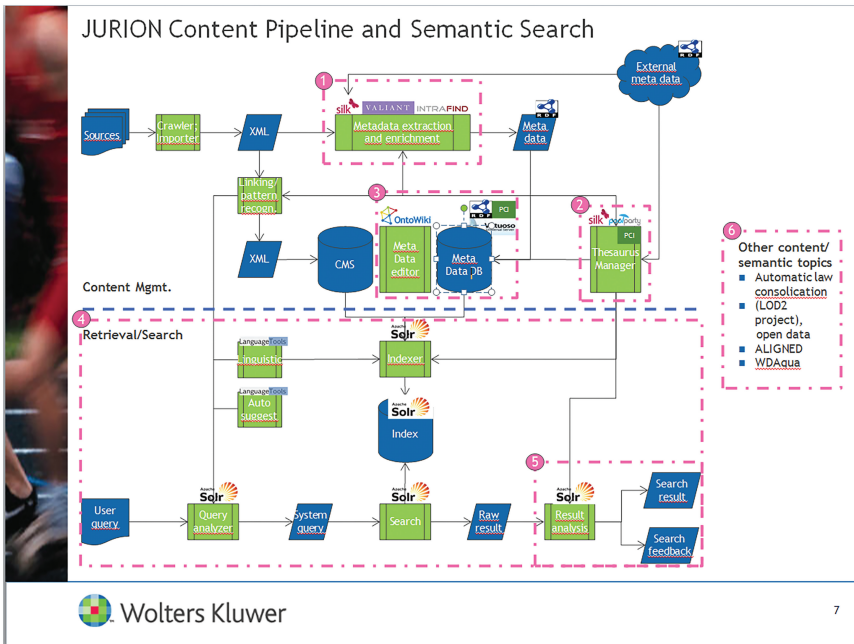


Figure 6.1 JURION Content Pipeline and Semantic Search.

By adopting the ALIGNED methodology and tools, software development and data processing pipeline maintenance will gain integrated governance mechanisms. These mechanisms will provide unified views of software and data engineering tasks enabled by linked enterprise Linked Data representations of both engineering teams. This will build on a common system specification language that produces and maintains links between data entities and code, executable code and program transformations that take account of how both systems co-evolve. The engineering process for both systems will be improved by the presence of new tools to integrate bug tracking and test results in both systems. ALIGNED methods and tools will streamline the processes for data acquisition, data processing, and data integration. These are all data curation activities that will be supported by workflows, model-driven generation of dataset-specific curation interfaces, automated data unit test generation, execution and reporting, data quality frameworks, and rule-based data integrity gateways. ALIGNED will enable JURION to address more complex business requirements that rely on tighter coupling of software and data.

6.1.3 Actors

Role	Description
CMS Expert	responsible for the technical correctness of process and data
Content Architect	responsible for the overall process and schemas
Legal Domain Expert	responsible for ensuring that legal data are correct
Legal Editor	responsible for editing legal information
Product Owner	wants the best possible product
Quality Manager	responsible for data quality assurance
Schema Expert	responsible for executing and documenting schema changes

The requirements on which the JURION use case was based are detailed in Appendix A.

Architecture

Based on the FP7 LOD2³ project, parts of the Linked Data stack have been deployed in JURION to handle data complexity issues (see Figure 6.2). The

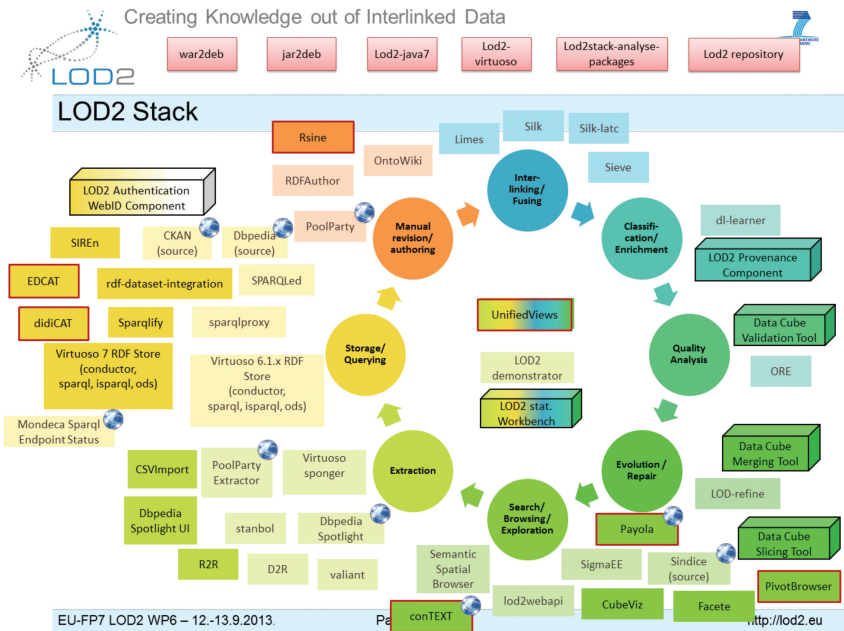


Figure 6.2 Distribution of the Linked Data stack components w.r.t. Linked Data Publishing cycle.

³<http://lod2.eu/Welcome.html>

FP7 LOD2 project aimed at developing novel, innovative Semantic Web technologies and also at the expansion and integration of openly accessible and interlinked data on the Web. WKD acted as a use case partner for these technologies, supported the development process of semantic technologies and integrated them to support the expansion of Linked Data. WKD also published some domain specific datasets.

The software development process and data life cycle at WKD are highly independent from each other and require extensive manual management to coordinate their parallel development, leading to higher costs, quality issues and a slower time-to-market. This is why the JURION use case in ALIGNED is located both within the software engineering as well as in the data processing area (see Figure 6.3).

In the initial prototype implementation, we aimed at the creation of a stable prototypical environment, in which we can start testing and evaluating implementations to encounter the current issues. In this first phase, we concentrated mainly on the enhancement of data quality and repair processes. Based on requirements, we started to work on data transformation issues and the improvement of data quality processes in PoolParty.

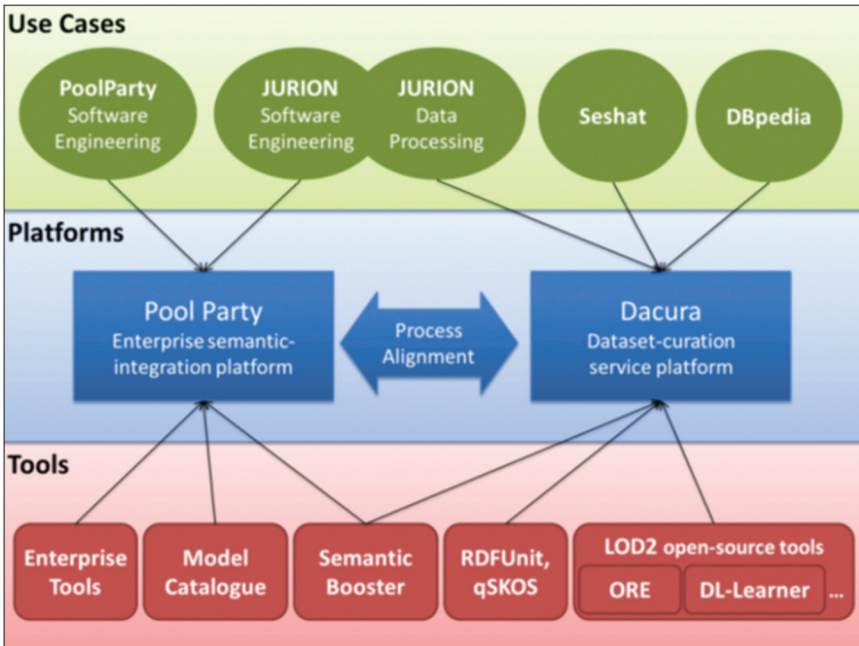


Figure 6.3 ALIGNED Use Cases.

6.1.4 Implementation

6.1.4.1 PoolParty notification extension

Development and maintenance of controlled vocabularies such as thesauri is mostly a manual and thus error-prone process. Especially in environments where multiple contributors are allowed to perform changes to the vocabulary, structural complexity increases, which makes it harder for individuals to maintain an overview. Furthermore, conflicting opinions arise and lead to inconsistent description, meaning and structure of the thesaurus' concepts. This problem is even more important when using software that allows for collaborative vocabulary development or publishing vocabularies as Linked Data.

Furthermore, maintaining an overview is not only necessary for thesaurus development, but also for curators responsible for datasets published as Linked Data on the Web serving various use cases. Users are allowed to change or add information (metadata) to existing data anytime. Therefore, errors can be introduced and hence manual review is required.

6.1.4.2 rsine notification extension

In order to address certain scenarios, the rsine⁴ notification service as well as its integration into PoolParty had to be extended. The changes incorporated into rsine's code dealt with support for persistence transaction and attaching rsine to receive notifications from other PoolParty repositories than the default vocabulary repository. Regarding the former change, multiple triple changes that are written into rsine managed triple store as one transaction are now combined and treated by rsine as one changeset. This allows for easier formulation of notification subscription documents and more robust notifications. On the PoolParty side, we added integration code that forwards changes to, e.g., the custom schema repository or the user account repository to rsine, so that it is also possible to get notified on schema and user account changes. However, this is just a temporary solution as we aim to get PoolParty working with a single repository and organise all other information in separate named graphs. Once this has been accomplished, also the rsine integration code can be simplified.

6.1.4.2.1 Results

To cover the most important scenarios, we implemented five new rsine notification subscription documents that enable notifications for

⁴See <https://github.com/rsine/rsine>

[WP5] Notification

wp5emailnotifier@lod2.eu

Gesendet: Mo 05.10.2015 10:41

An: Eck, Katja

A custom scheme named 'test scheme' has been created by user . You receive this notification because of subscription 'http://example.org/aligned/new_scheme' (Notification custom schema creation)

Figure 6.4 Notification message.

- Creation of a new custom class
- Creation of a new custom schema (see Figure 6.7)
- Deletion of a custom schema
- Creation of a new user account
- Creation of a new project

The other scenarios can be covered with similar subscriptions. Up to now, the details covered in the notification messages sent out to the users cannot cover information like

- who (username) created a custom schema or user account, and
- the name of the newly created schema

The reason for this is that (i) this information is not available in the persisted data or (ii) the repository holding the data is not available for querying through a SPARQL endpoint. Figure 6.4 shows a sample notification message.

6.1.4.3 RDFUnit for data transformation

As part of the core, CMS tasks within JURION each WKD XML document that is checked-in through internal workflow functionality and is converted to RDF based on the Portal Content Interface (PCI) ontology. The PCI ontology is a proprietary schema that describes legal documents and metadata in OWL. Due to change requests and new use cases for the RDF metadata in the ontology, the conversion logic or both the conversion logic and ontology need amendments. In these cases, we need to ensure that the RDF data that are generated from the WKD XML documents still comply with the PCI ontology for quality assurance.

Current Situation

As a gatekeeper to avoid loading flawed data into the triple store, each result of the conversion from WKD XML into PCI RDF is sent to a proprietary dedicated Validation service that inspects the input and verifies compliance with the ontology. This approach assures that the conversion results

are verified but comes with some major issues. The three most important ones are:

- The current service can only process larger data packages. This makes error detection on single data units quite difficult and one error blocks the whole processing pipeline
- the service is a SOAP-based Web service that works asynchronously with many independent process steps, which imposes high complexity on its usage
- it depends on other services and requires permanent network access and therefore is potentially unstable

To improve these issues, we want to implement unit test scenarios that can be run directly coupled to the conversion project development environment (this project hosts XSLT logic to convert WKD XML into PCI RDF). The tests should be run both automatically on every change in the project, but also be able to be manually triggered. Tests should be easily extendable and expressive enough to easily spot issues in the conversion process. The feedback loop should be coupled as tight as possible to the submitted change.

Implementation

To allow comparable and reproducible test results with suitable execution time, a number of WKD XML reference documents have been selected, against which the actual conversion into PCI RDF is executed and each resulting RDF dataset is verified individually.

The prototyped solution (see Figure 6.5) integrates RDFUnit as the core driver of the tests. The integration is currently based on auto generated tests, which are generated from a current version of the PCI ontology every time the test suite is run.

It also integrates seamlessly into the general development toolchain. Any change in the conversion project automatically leads to an entire build of the project including validation. As the test suite is integrated in the underlying standard test mechanisms, a developer can trigger this test chain manually on his local workstation to retrieve direct feedback at any time.

As a proof of concept RDFUnit's test results (the validation model based on the Test-Driven Data Validation Ontology⁵) linked to this test is stored into Virtuoso triplestore to enable future analysis/reviews of historical data.

⁵See <http://rdfunit.aksw.org/ns/core>

Results

Each of the test results manifests in the validation model, which is based on RDFUnit’s Test-Driven Data Validation Ontology. As we currently rely on RDFUnit’s auto generators, all statements are spotted that outcast rules that have been derived/interpreted from the ontology. These are especially cardinality and domain/range violations.

In any case, a summary of the test results is presented to the user. As this is always in the context of a concrete RDF-dataset (in the form of a file) one can immediately spot issues on the exact resource, which avoids unnecessary lookups and helps to identify the defective part of the conversion.

The integration of RDFUnit into the development cycle and build pipeline (see Figure 6.5) enabled the following possibilities that were entirely missing before:

- run automated tests based on the ontology
- steadily monitor project health
- capture metrics

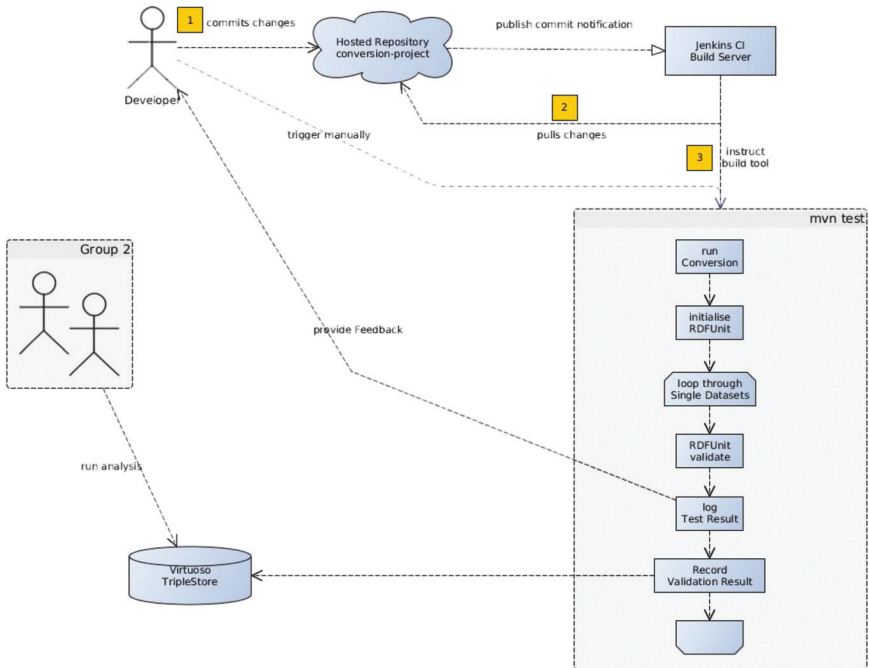


Figure 6.5 Transformation process with RDFUnit.

In the past, most issues aroused after the changes to the conversions have been released without proper and reliable testing – as this was only possible in manual developer tests. Moving forward, we can make sure that reproducible tests are run with each change especially before releases. Tests and tested documents can be easily extended to increase coverage of corner cases.

Figures 6.6 and 6.7 shows some of the test results, which can easily be stored and used on a regular basis in current and future QA reports.

Early and quick feedback on changes to the project are very valuable to assure that the project is in good health and existing functionality meets the defined expectations. Good coverage with automated tests prevents bugs from slipping in released functionality which may have bad side effects on other parts of the system.

RDFUnit enables possibilities but still needs a tighter integration as a library with our existing toolchain to improve reporting capabilities and make its feedback even more useful.

RDFUnit proves as being very useful and will be a fixed component of the operational tech stack within WKD JURION from now on.

We will provide further requirements to improve RDFUnit’s integration into our development pipeline. At a later point in time, we will utilise RDFUnit to enable monitoring the existing data store to implement quality assurance on operational side.

```
target/test-classes/junit7523938743608749278/output/baulast_13211.meta.rdf

[ERROR] http://wolterskluwer.de/ceres/wk-
de/lexdb/181634/baulast_13211#Hinweis01bea53a31ad369b9dabd6a4704230ef
. Cardinality of http://wolterskluwer.com/ceres/concept-v1.0/anchorId
different from 1 (is 0) for type http://wolterskluwer.com/ceres/content-
warehouse-v1.0/BlockAnchor
. Cardinality of http://wolterskluwer.com/ceres/concept-v1.0/anchorId
different from 1 (is 0) for type http://wolterskluwer.com/ceres/concept-
v1.0/Anchor

[ERROR] http://wolterskluwer.de/ceres/wk-de/lexdb/181634/baulast_13211
. http://wolterskluwer.com/ceres/wk-
de/referenceInformation.ChapterReference does not contain a literal value
(http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral)
. http://wolterskluwer.com/ceres/wk-
de/referenceInformation.ChapterReference has rdfs:domain different from:
http://wolterskluwer.com/ceres/ltr-v1.0/ReferenceInformation
. http://wolterskluwer.com/ceres/wk-de/searchTuningKeyword has rdfs:domain
different from: http://wolterskluwer.com/ceres/concept-
v1.0/InformationClass
. http://wolterskluwer.com/ceres/content-warehouse-
v1.0/isDocumentInstanceOf has rdfs:domain different from:
http://wolterskluwer.com/ceres/concept-v1.0/FileResource
```

Figure 6.6 RDFUnit results.

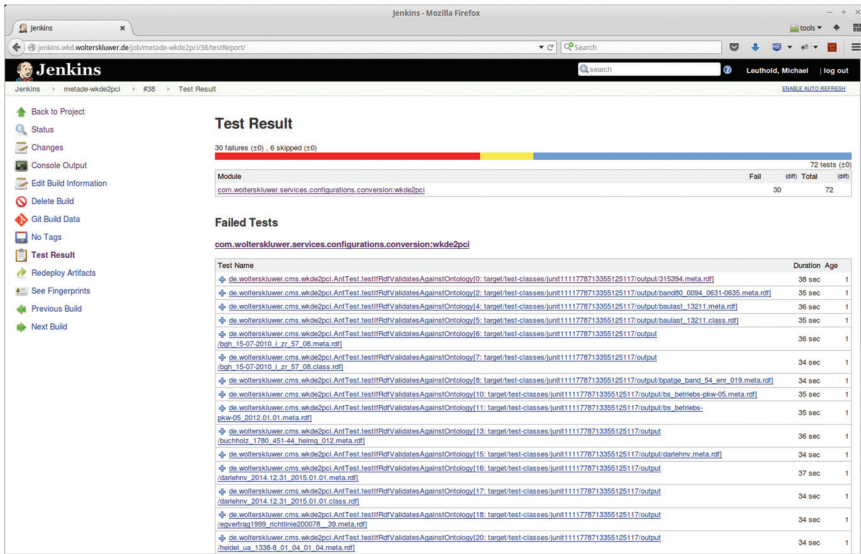


Figure 6.7 Jenkins-CI Test Report.

6.1.4.4 PoolParty external link validity

WKD document metadata and controlled vocabularies are linked to several external sources. These sources are mainly DBpedia⁶ and thesauri like Eurovoc⁷, Thesoz⁸ or STW⁹. On a larger scale, we plan to include more of these kinds of sources to connect with additional internal and external data for the enhancement of several services. To control the process of change and to evaluate what kind of effects this can have on the quality of data, we want to control changes of Linked Data that can cause problems.

In addition to the validity of external links, we also aim to monitor the validity of internal links between different projects and datasets as also internal WK sources will need validity control.

Current Situation

Currently, we have no effective overview over the validity of linked sources. This causes, for example, frontend problems in the published vocabularies (see Figure 6.8). Currently, the only way to evaluate the quality is to analyse

⁶See <http://de.dbpedia.org/>

⁷See <http://eurovoc.europa.eu/drupal/>

⁸See <http://www.gesis.org/en/services/research/thesauri-und-klassifikationen/social-science-thesaurus/>

⁹See <http://zbw.eu/stw/version/latest/about.en.html>

execution	source	errors	failed	testsRun	testsSuccessful	timeout	totalViolations	start	end
http://vd.umiami.akvo.org/data/results/c3b6c1ba-Zae4-11b2-8084-d4bed9680e44	rdfmmit-0074d1491-30c0-4ea3-8988-80591bb1c3d	0	27	2922	2895	0	244	2015-07-20T12:25:08.749Z	2015-07-20T12:25:17.876Z
http://vd.umiami.akvo.org/data/results/a8f72716-Zae0-11b2-8084-d4bed9680e44	rdfmmit-00948713-0bd4-4016-81e-0283384c-90a	0	26	2922	2896	0	236	2015-09-17T14:38:45.202Z	2015-09-17T14:39:23.488Z
http://vd.umiami.akvo.org/data/results/59c21293-Zae5-11b2-8072-843ab6a105c	rdfmmit-00a0affb-965c-47c9-ab11-b0477e4136d0	0	34	2922	2888	0	303	2015-07-23T09:23:41.883Z	2015-07-23T09:23:51.479Z
http://vd.umiami.akvo.org/data/results/736a2713-Zae4-11b2-8084-d4bed9680e44	rdfmmit-00d1dc06-3b65-4a0b-a26e-c077602978ca	0	11	2922	2911	0	234	2015-07-17T12:41:54.213Z	2015-07-17T12:42:05.511Z
http://vd.umiami.akvo.org/data/results/a9f74908-Zae0-11b2-8084-d4bed9680e44	rdfmmit-00eace7c3-8618-46c1-7d0d-c40e49367648	0	8	2922	2914	0	218	2015-09-17T14:54:29.594Z	2015-09-17T14:55:03.412Z
http://vd.umiami.akvo.org/data/results/a2958ec3-Zae0-11b2-8084-d4bed9680e44	rdfmmit-01b17591e-2361-60df-4e0f-64e9815e738	0	29	2922	2893	0	292	2015-09-17T13:06:61.176Z	2015-09-17T13:06:26.823Z
http://vd.umiami.akvo.org/data/results/a564be11-Zae0-11b2-8016-d4bed9680e44	rdfmmit-01e75c1a-33c4-4385-9a1e-863778876527	0	9	2922	2913	0	228	2015-09-17T13:24:59.230Z	2015-09-17T13:25:26.056Z
http://vd.umiami.akvo.org/data/results/a65906c5-Zae4-11b2-8043-d4bed9680e44	rdfmmit-026f170d-39e8-417c-beaf14524f0d4b3e	0	27	2922	2895	0	244	2015-07-20T12:36:19.605Z	2015-07-20T12:36:30.724Z
http://vd.umiami.akvo.org/data/results/a53327ac-Zae0-11b2-801e-d4bed9680e44	rdfmmit-02874511-970e-4ecf-42c8-806e3f08d0c	0	8	2922	2914	0	218	2015-09-17T12:54:46.715Z	2015-09-17T12:55:12.986Z
http://vd.umiami.akvo.org/data/results/a52fd411-Zae0-11b2-8016-d4bed9680e44	rdfmmit-031d4dc2-ec3e-405e-4319-0088653e233	0	34	2922	2888	0	640	2015-09-17T13:16:35.189Z	2015-09-17T13:17:03.117Z
http://vd.umiami.akvo.org/data/results/38702728-Zae4-11b2-8007-d4bed9680e44	rdfmmit-03269411-20ed-4e14-b010-73608d1f08c	0	13	2922	2909	0	223	2015-08-31T12:18:11.336Z	2015-08-31T12:18:44.012Z
http://vd.umiami.akvo.org/data/results/58c45739-Zae5-11b2-8022-843ab6a105c	rdfmmit-03b446c5-050-4b45-83e2-48591fd0386	0	11	2922	2911	0	227	2015-07-23T09:24:01.972Z	2015-07-23T09:24:10.794Z
http://vd.umiami.akvo.org/data/results/cd113705-Zae4-11b2-8043-d4bed9680e44	rdfmmit-03c3c3ab-395e-408e-8d74-869e41eac0f	0	34	2922	2888	0	303	2015-07-20T12:33:58.039Z	2015-07-20T12:34:09.576Z
http://vd.umiami.akvo.org/data/results/5a377ec5-Zae5-11b2-8084-d4bed9680e44	rdfmmit-04d02032-0e27-4a11-8605-277996c1d5f	0	30	2922	2892	0	287	2015-07-23T09:40:06.713Z	2015-07-23T09:40:16.573Z
http://vd.umiami.akvo.org/data/results/f0e97102-Zae4-11b2-8084-d4bed9680e44	rdfmmit-04e33241-1a4f-4560-8313-5ae79109278	0	19	2922	2903	0	230	2015-07-20T12:50:35.183Z	2015-07-20T12:50:44.932Z
http://vd.umiami.akvo.org/data/results/59bb8840-Zae4-11b2-800e-d4bed9680e44	rdfmmit-05341384-c0ca-4028-b01e-6e58a36f1191	0	31	2922	2891	0	311	2015-08-28T13:08:22.650Z	2015-08-28T13:08:35.082Z
http://vd.umiami.akvo.org/data/results/5a2a8e5c-Zae5-11b2-8084-d4bed9680e44	rdfmmit-05f08039-852e-4318-8b7e-77e8a4285dc1	0	28	2922	2894	0	238	2015-07-23T09:35:04.626Z	2015-07-23T09:35:16.455Z
http://vd.umiami.akvo.org/data/results/f06a3542-Zae4-11b2-8084-d4bed9680e44	rdfmmit-0702530f-6926-4370-afcd-858446e87498	0	10	2922	2912	0	220	2015-07-20T12:50:45.350Z	2015-07-20T12:50:55.196Z
http://vd.umiami.akvo.org/data/results/a3605d7a-Zae0-11b2-8081-d4bed9680e44	rdfmmit-009640e4-a378-4788-9153-a5a3e88288c	0	29	2922	2893	0	285	2015-09-17T14:55:38.112Z	2015-09-17T14:56:12.094Z
http://vd.umiami.akvo.org/data/results/a396a33c-Zae0-11b2-8081-d4bed9680e44	rdfmmit-08953201-a0cd-4c2e-a027-13c1557eac0	0	34	2922	2888	0	244	2015-09-17T14:52:11.240Z	2015-09-17T14:52:45.756Z
http://vd.umiami.akvo.org/data/results/939c2db0-Zae4-11b2-8073-d4bed9680e44	rdfmmit-08da3c8e-01bc-49ec-81b1-329e6f66b0d	0	9	2922	2913	0	231	2015-08-31T12:55:16.861Z	2015-08-31T12:55:50.690Z
http://vd.umiami.akvo.org/data/results/97e7d77c-Zae0-11b2-8022-843ab6a105c	rdfmmit-09d4783b-4d69-480b-8303-4e8f06917384	0	26	2922	2896	0	242	2015-09-17T06:42:49.918Z	2015-09-17T06:43:17.442Z
http://vd.umiami.akvo.org/data/results/c5087624-Zae4-11b2-8043-d4bed9680e44	rdfmmit-09ab88dc-e0f0-4e4f-a022-c84171b2ac0	0	37	2922	2885	0	644	2015-07-20T12:31:21.974Z	2015-07-20T12:31:35.587Z

Figure 6.8 Validation Data stored for Total Analysis.

the frontend representations of the linked sources or to follow a link to detect a missing source. There is in general no process in place to control the validity of external sources. Figure 6.9 shows a sample defect.

Implementation

To check the validity of external links, we use the same technique as qSKOS.¹⁰ All URIs used in the vocabulary that do not point to the local host are dereferenced and the remote server’s response is checked. If the HTTP status code is 200, the link is considered valid. In case redirects occur, they are followed properly. All other responses are to be classified as invalid.

Results

URI checking can be invoked from the PoolParty user interface in the current experimental version. The result overview (see Figure 6.10) shows the URIs of the violated links and the total number of checked links as well as the number of violated links. The quality manager can use these links to change or delete the respective relations.

However, since each URI gets resolved and duplicate URIs are not omitted, this process can take a lot of time. In future versions of PoolParty, we

¹⁰See <https://github.com/cmader/qSKOS>

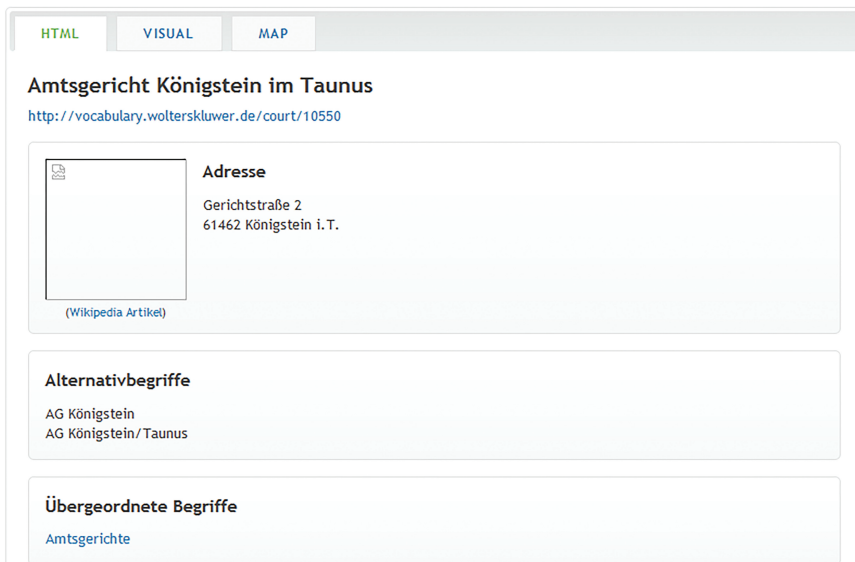
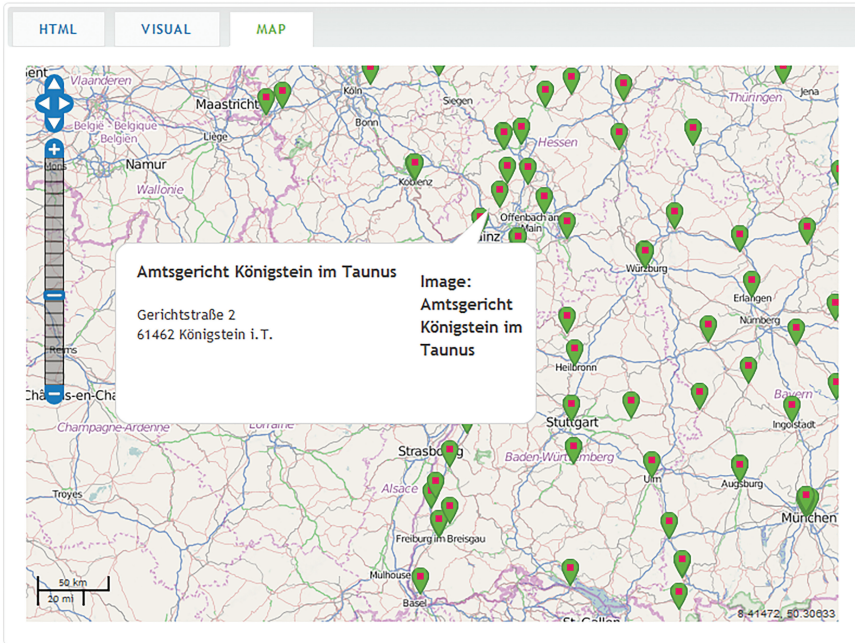


Figure 6.9 Example defect: the Image file of the external source does not exist anymore¹¹.

¹¹ See frontend <http://vocabulary.wolterskluwer.de/court/10592.html>

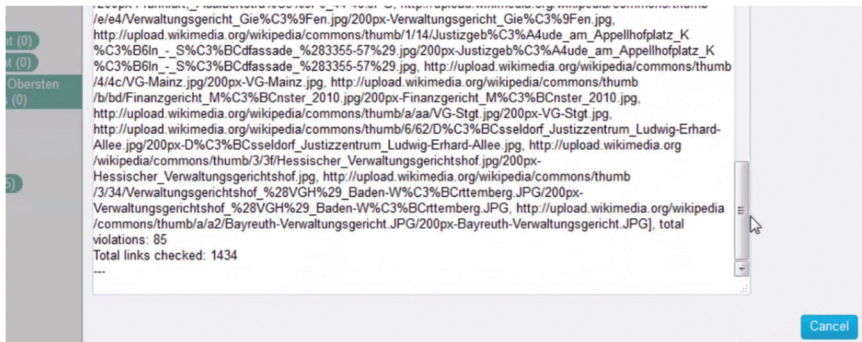


Figure 6.10 Validation Results.

will investigate ways of running these kinds of checks in the background and notify users on the results.

6.1.4.5 Statistical overview

As we are integrating more and more controlled vocabularies and custom schemas in the metadata management tool PoolParty, we are in need of solutions that give an overview of existing relations between projects and external data and schemas. Besides, the number of user roles is growing so that we need a solution that enables a best overview for a number of different users with different purposes. By different queries and enhancements, we want to get an impression about the relations between projects and the usage of specific custom schemas.

Current Situation

Connections of projects and schemas are not easily traceable. Owners of vocabularies need to document everything so that others can also understand the projects and its relations and possibilities. Without this documentation, it is hard to analyse the different projects. Within the tool, the user can only analyse the individual concepts for relations to investigate any relations with schemas. For linking to other projects it is possible to get a list of links. This list does not provide the number of links and specific numbers for different kinds of linking. These figures need to be searched manually.

Implementation

We currently implemented two different kinds of statistical metrics and integrated them into the PoolParty UI (i) checking for external links validity

and (ii) links to other PoolParty projects on the server. These metrics differ in the methodology they are evaluated. Checking the validity of external links cannot be done using SPARQL and requires external tool support (e.g., Java code, see section 6 on external link validity). Reporting links to PoolParty projects can be achieved in a similar way than checking for data consistency violations. Each statistical property can be formulated as a SPARQL query, which is executed on the relevant project data, i.e., the current project data and metadata as well as all linked project data and metadata.

Results

The checking of project relations can be invoked from the PoolParty user interface in the current experimental version. The results (see Figure 6.11) show the kind of used relations, the frequency of these relations, the detailed list of linked resources and the total number of linked resources.

This way users can check how and to which extent projects are related to each other and they get an overview of used relations.

6.1.5 Evaluation

The Jurion Use case is split into two sections within the ALIGNED project: (1) the Jurion platform, and (2) the Jurion IPG tool. The developments concerning the Jurion platform took place in the first half of the project, based on the respective categories of measurement and will be repeatedly described here for completeness.

For the prototype of the JURION platform use case, we focussed on the data development processes.

The ALIGNED tools that were used for this prototype are RDFUnit and PoolParty. We had four major features for the initial prototype.



Figure 6.11 Statistical checks.

- RDFUnit for Data Transformation
- Notification Service in PoolParty
- Project Linking Statistics in PoolParty
- ELV in PoolParty

The methods of collection are divided into three categories, namely productivity, quality, and agility, as follows in Figure 6.12.

Tasks	Comment	Productivity (Prototype testing)	Quality (Prototype Testing, expert evaluation/ interviews)	Agility (expert evaluation)
RDF Transformation	quality test of data transformed from XML to RDF	Time Measurement for Quality Checks Time Measurement for Error Detection Need for Manual Interaction	number of detected error categories test coverage expert evaluation	Time to include new constraints/adapt the testing to new requirements
Notification	notification about predefined changes	Number of Scenarios Time Measurements Usefulness	Notification completeness expert evaluation	Time to include new constraints/adapt the testing to new requirements User roles that can modify Notifications Time to configure a new Notification Integration of a customized Notification Configuration Time to configure new requirements
Statistics	Statistics about relations of projects	time detected links	usability aspects result consistency	Detection Issues Integration of Statistics Time to configure new Requirements Extension
External links	quality of external links	checked links violations time	usability aspects expert evaluation/inter correctness of results	Scope of External Link Checks Integration of Internal Link Checks Time to configure new Requirements Extension

Figure 6.12 JURION: Overview.

The evaluation of the prototype showed clearly that during the Jurion prototype development, we have achieved our aim to improve the productivity and quality of data processes within the data life cycle. With the presented features, these improvements could be made visible. Performance and quality/error rates of the test results were satisfactory. Nonetheless, evaluation outcomes suggest further improvements are possible, especially with regard to usability, performance, integration of functionalities and required details that are not yet fully optimised.

6.1.5.1 Productivity

In summary, the productivity of data processes is clearly improved by the Jurion prototype. The data transformation service enables a testing that points directly to the detected error source and improves the bug fixing process this way. The notification service provides notifications as soon as an action is executed. This is a helpful tool to ensure quality analysis and data monitoring. Nonetheless, there needs to be a solution that helps to send the notification precisely where it is needed to avoid spamming. The statistics and ELV functionalities can help to save much time by replacing time-consuming manual work with efficient data overviews.

6.1.5.2 Quality

Concerning the quality of the prototype functionalities, the results are very satisfying. For notifications and ELV, there are only few issues. For the data transformation with RDFUnit and the statistics part, there needs to be further investigation to enable comprehensive and extensive data testing results. Usability issues need to be tackled in all the features for a better operational implementation. As this is only an initial prototype, usability was less of a priority.

6.1.5.3 Agility

The testers' feedback for agility of features is quite positive. The agility of RDFUnit is seen as satisfying as the automated service allows the implementation of new requirements easily. With regard to notification, adaptations are dependent on the specific notification use case and the respectively available data. In the same way, the agility of statistics feature is highly dependent on the availability of required underlying data.

ELV has a reasonable agility and is planned to be done by an external application to address performance issues.

The evaluation of the Jurion tasks was done in an early phase of the project, based on an earlier evaluation approach. We will analyse one of the tasks based on the latest suggested method to show the adaptability of the test results for this approach. Task 4 ELV service serves as a good example for this analysis.

6.1.5.4 Measuring overall value

JURION is a legal information platform that merges and interlinks over one million documents of content and data from diverse sources such as national and European legislation and court judgements, extensive internally authored content and local customer data, as well as social media and Web data (e.g., from DBpedia). The JURION development teams add value through applications for personalisation, alerts, analysis and semantic search. Revenue is generated by customers paying for the platform content and related services.

PoolParty serves as the metadata management tool of controlled vocabularies that are used for specific search functionalities and the development of further functionalities in applications. The ELV is a PoolParty functionality.

ELV is a new feature that evaluates the links to external sources and informs the user in case the sources are not available anymore. Previously, it was only possible to check the links manually in random samples. Therefore, it provides a fast and efficient curation service to guarantee an error-free linking to external sources. A measure of value could be curation cost of maintaining a given quality of service as measured by revenue. The saving of time needed for the error detection is the most important parameter for this calculation.

6.1.5.5 Data quality dimensions and thresholds

Data accuracy, completeness and consistency are essential for this task. Jurion Customers pay for the curated information and related services so that high-data quality is a major requirement. Data accuracy was analysed in the evaluation by analysing the errors – 100% of the found errors have been data inconsistencies. In average, 81% have been outdated links, we were looking for. Nineteen per cent have been unexpected inconsistencies that exceeded

our expectations. Completeness was checked via mapping the errors that were found manually, against the system results. All manually detected issues have been detected by the links.

6.1.5.6 Model agility

As the functionality is embedded in Poolparty, the assessment of the Poolparty use case is also valid in this case. With regard to the functionality, the configuration of new requirements for the ELV is possible. Determining which URIs should be resolved can be done either with the methods SPARQL provides or within the Java resolution algorithm. In each case, the effort for change is low, allowing for agile reaction on changed requirements. However, changes to the current configuration require recompilation and redeployment of PoolParty.

6.1.5.7 Data agility

As the functionality is embedded in Poolparty, the assessment of the Poolparty use case is also valid in this case. Based on the pattern to detect URI patterns for links to be checked, the solution can also be used for (or constrained to) “internal” links. Therefore, appropriate methods must be evaluated.

6.1.6 JURION IPG

6.1.6.1 Introduction

The Jurion IPG system is a commercial intelligence system, providing a means for business contractors to perform due-diligence queries, serving historical data about companies and their relationships with other companies, responsible individuals, and business documents. As a reliable provider of credibility and financial information for over five million entities, the integrity and consistency of the data is of vital importance, and increasingly hard to manage at scale. In this use case, we are deploying the ALIGNED tools to find problems in the existing data and to improve the integrity of data submitted in the future. ALIGNED tools are also helping increase the scope of the data, by enabling the linking of data stored within the system to external related datasets.

Figure 6.13 shows the flow of content through components of the system. Source data are manually imported or acquired through crawling non-formatted data sources, and pushed into a relational data store. Metadata is extracted and enriched, before being entered into a separate RDF data store.

JURION Content Pipeline

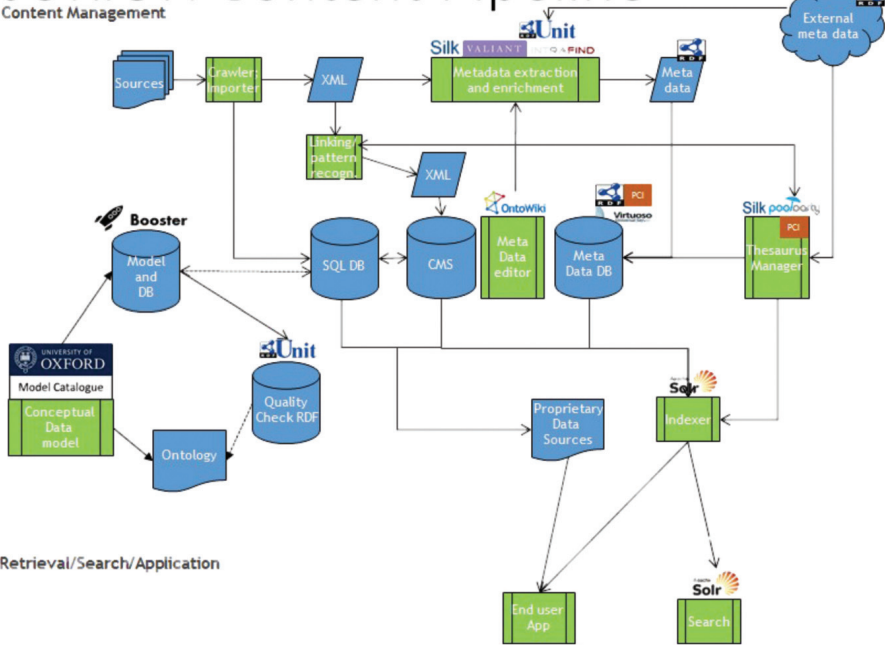


Figure 6.13 JURION Content Pipeline, showing ALIGNED tools integrated with existing functionality and datasets.

The schema for the relational data store is versioned and updated through an instance of the Model Catalogue; and data integrity is maintained through Semantic Booster-generated stored procedures. This relational data may also be viewed in an RDF format, where the RDFUnit tool may be used for further data validation. External metadata is managed through use of the SWC’s PoolParty thesaurus manager and linked with the RDF representation of the core dataset. Existing end user interfaces to the data will be supplemented with an administrative interface automatically generated by the Semantic Booster tool.

New software integration points can be found where ALIGNED tools interact or communicate. In particular, such interactions occur between the Model Catalogue and Semantic Booster, where Booster models are generated from the Model Catalogue, and where metadata in the catalogue is used to supplement the end user interfaces. Further integration is between Booster and RDFUnit, where the D2RQ tool is used to help convert relational

data into RDF: the configuration for this may again be parameterised by a transformation of the model in the Catalogue. The ALIGNED vocabularies are used to standardise these interactions.

Use Case

Figure 6.14 shows the problem space and more specifically the complexity of the JURION IPG system. The utility of the JURION IPG system is dependent on the maintenance and evolution of a large, semantically consistent dataset. Huge amounts of daily processed data originally from pdf sources; and maintenance through a proprietary, obsolete CMS makes the IPG case extremely suited as an ALIGNED use case. Business value of the system is dependent on the maintenance and evolution of a large, semantically consistent dataset. The overall goal is to ensure the quality of the system used to enter and maintain the data and to improve the value by linking to external datasets. To provide this solution by implementing ALIGNED tools, we used in parallel two approaches including Semantic Booster first and Dacura afterwards. For the purpose of the Jurion IPG use case, we have chosen to concentrate on a number of key critical concerns:

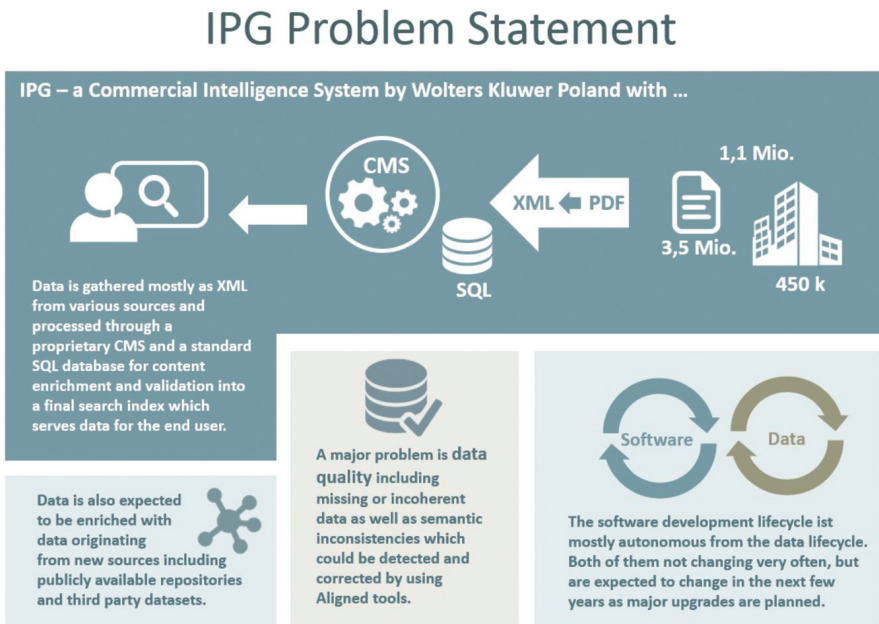


Figure 6.14 IPG problem statement.

- The use of Semantic Booster will allow a wider range of semantic integrity constraints and business rules to be applied to the data upon entry, ensuring high-quality data. The automatic data-migration tools provided with Booster will minimise the impact of upgrading and evolving the underlying data model whilst maintaining data consistency.
- The administrator interface in the IPG system currently requires manual development each time the database changes; increasing the cost of evolving the data store. The model-driven Booster default interface can be used: either in its entirety, or components reused to save development effort. (Figure 6.14)
- The existing data store is currently stored in a relational format. Whilst Booster can help enforce a range of integrity constraints, there are some consistency checks which would be more reliably performed using RDF reasoning; some additional constraints may be enforced in a less severe manner: not enforced globally but treated on a case-by-case basis. We will use the existing D2RQ¹² tool to convert data stored within a Booster database into RDF format, making it available to the RDFUnit testing tool. D2RQ is a platform and language for accessing standard relational data, as that found in Booster, as triples. It is the basis from which the R2RML¹³ W3C standard was developed. In D2RQ, each element of a Linked Data schema can be mapped to data from a relational database, using standard SQL queries, embodied in a mapping file in the D2RQ formalism. The additional testing and monitoring this enables will also provide insight into productivity and quality gains through use of the ALIGNED tools.
- Semantic integrity of the data can be compromised by a lack of understanding of the model. Here the Model Catalogue can be used to provide accurate descriptions of data fields, including those from linked external data sources. Such descriptions can aid correct data entry, and permit additional reuse of the data within the organisation. The Catalogue will also serve as a provider of models to the generated tools, and an environment where new versions of the data model can be created and evolved.

In the ALIGNED use case, the IPG domain model is edited and versioned within the Model Catalogue: Figure 6.15 shows a screenshot including a subset of the model. The model can be used as the foundation for a model in Semantic Booster, but the Catalogue is also able to generate

¹²<http://d2rq.org/>

¹³<https://www.w3.org/TR/r2rml/>

Entity Draft
Data Class
 Last Update: 2016-08-17 20:24:22

Description	An entity - either a company or an individual person
Parent Hierarchy	Wolters Kluwer IPG Model
Classifications	

DataElements
Metadata
Annotations
Links
Change History

DataElements +

Name	Data Type	Description
Name	String	The registered name of a company, or the firstname, surname... more
Foreign	Boolean	Whether a company is registered outside of Poland, or a... more
Notes	Boolean	Additional text notes about the company or person.
Documents	Boolean	A set of files about this entity.
Source of Relationships	Set(Relationship)	Relationships where the source is this entity.
Target of Relationships	Set(Relationship)	Relationships where the target is this entity.

5 10 20 50

Figure 6.15 Screenshot of a subset of the IPG model in the Model Catalogue.

documentation files, data interchange specifications (such as in XML), and other useful system components.

The Booster model may be further edited within the Eclipse-based IDE (Figure 6.16) to extend concepts with further business rules and update methods. The Booster generation system is then used to generate a database with stored procedures for updates, a programmatic API, and a Web-based administrative interface (Figure 6.17).

The data within the Booster system can be extracted in RDF format using the D2RQ tool. These RDF data are now suitable for linking to external datasets, or further reasoning. The RDFUnit tool can be used for

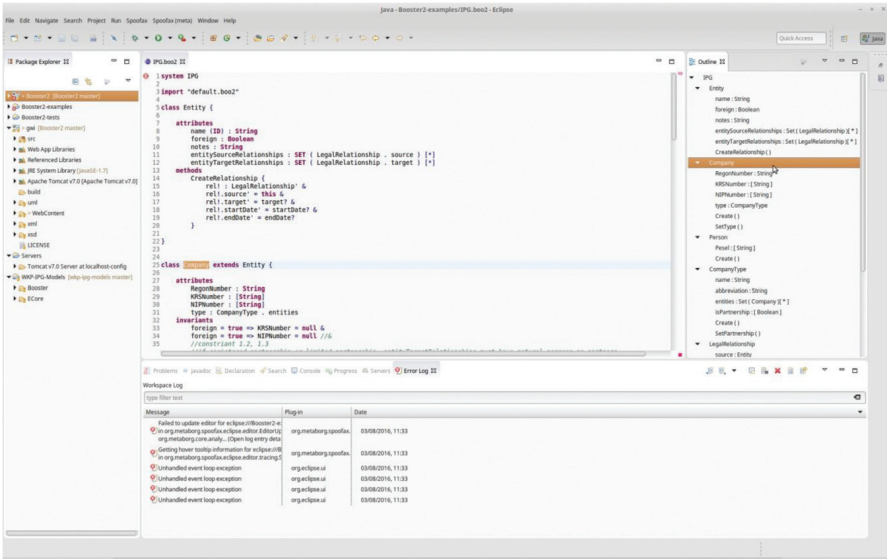


Figure 6.16 The Eclipse-based Booster tool.

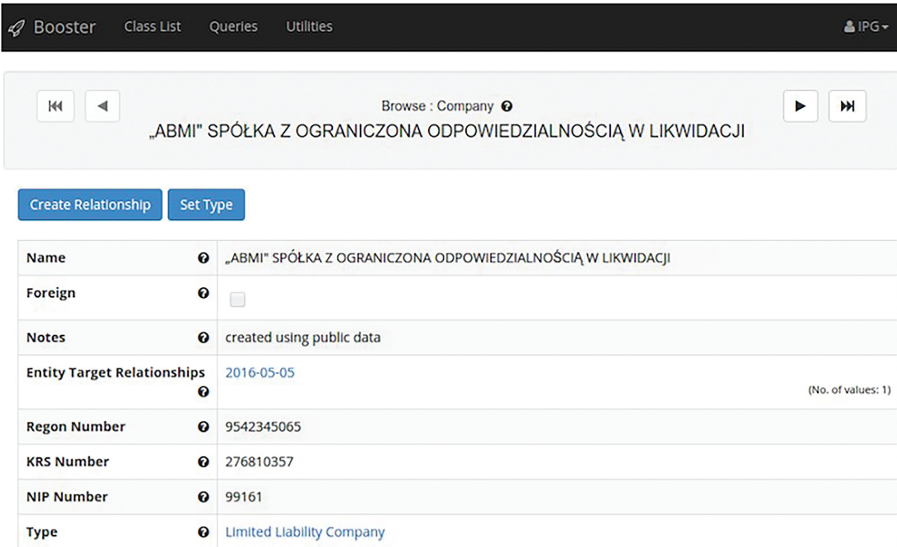


Figure 6.17 Screenshot of the Booster administrator interface for the JURION IPG system.

performing extra validity checks on these data (Figure 6.18) – checks that might be hard to describe or perform within a relational framework, or properties concerning relationships with external data.


```

42 <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d/8d2adb57-2b2c-11b2-80c3-dc0e1faf74d>
43   a                                     rut:TestCaseResult , rut:StatusTestCaseResult , rut:AggregatedTestResult ;
44   dcterms:date                         "2016-07-11T07:58:58.296Z"^^xsd:dateTime ;
45   dcterms:description                  "https://w3id.org/igp#hasLegalRelationshipKind does not contain a literal value (htt
46   rut:resultCount                       0 ;
47   rut:resultPrevalence                 -1 ;
48   rut:resultStatus                     rut:ResultStatusSuccess ;
49   rut:testCase                          rutt:aligned.cs.ox.ac.uk_schemas_igp.generated.ttl-OBJFUNC-39400effb8c242f0c91e3b62
50   rut:testCaseLogLevel                 rlog:ERROR ;
51   prov:wasGeneratedBy                  <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d> .
52
53 <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d/8d2adb4d-2b2c-11b2-80c3-dc0e1faf74d>
54   a                                     rut:TestCaseResult , rut:AggregatedTestResult , rut:StatusTestCaseResult ;
55   dcterms:date                         "2016-07-11T07:58:58.378Z"^^xsd:dateTime ;
56   dcterms:description                  "https://w3id.org/igp#hasKRSNumber has rdfs:domain different from: https://w3id.org/
57   rut:resultCount                       0 ;
58   rut:resultPrevalence                 -1 ;
59   rut:resultStatus                     rut:ResultStatusSuccess ;
60   rut:testCase                          rutt:aligned.cs.ox.ac.uk_schemas_igp.generated.ttl-RDFSDOMAIN-4fef956f9f7ec5ed20640f
61   rut:testCaseLogLevel                 rlog:ERROR ;
62   prov:wasGeneratedBy                  <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d> .
63
64 <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d/8d2adb70-2b2c-11b2-80c3-dc0e1faf74d>
65   a                                     rut:AggregatedTestResult , rut:StatusTestCaseResult , rut:TestCaseResult ;
66   dcterms:date                         "2016-07-11T07:58:58.282"^^xsd:dateTime ;
67   dcterms:description                  "https://w3id.org/igp#hasEndDate does not contain a literal value (http://www.w3.org
68   rut:resultCount                       0 ;
69   rut:resultPrevalence                 -1 ;
70   rut:resultStatus                     rut:ResultStatusSuccess ;
71   rut:testCase                          rutt:aligned.cs.ox.ac.uk_schemas_igp.generated.ttl-OBJFUNC-86dafa59272657289d4b04f6
72   rut:testCaseLogLevel                 rlog:ERROR ;
73   prov:wasGeneratedBy                  <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d> .
74
75 <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d/8d2adb4f-2b2c-11b2-80c3-dc0e1faf74d>
76   a                                     rut:StatusTestCaseResult , rut:TestCaseResult , rut:AggregatedTestResult ;
77   dcterms:date                         "2016-07-11T07:58:58.403Z"^^xsd:dateTime ;
78   dcterms:description                  "https://w3id.org/igp#legalRelationshipType has different range from: https://w3id.o
79   rut:resultCount                       0 ;
80   rut:resultPrevalence                 -1 ;
81   rut:resultStatus                     rut:ResultStatusSuccess ;
82   rut:testCase                          rutt:aligned.cs.ox.ac.uk_schemas_igp.generated.ttl-RDFS RANGE-807576591b580938ed3a7f5
83   rut:testCaseLogLevel                 rlog:ERROR ;
84   prov:wasGeneratedBy                  <kurn:uuid:8d2adb48-2b2c-11b2-80c3-dc0e1faf74d> .

```

Figure 6.18 Results of using the RDFUnit tool against data from a Semantic Booster database.

Dacura provides several services to software engineers developing software that utilises the data curated by the system. These include reliable access to data models, change notifications and the automatic production of simpler formats, which are more familiar to developers.

In order to verify the effectiveness of both approaches – Semantic Booster and Dacura – we created a list of unsolvable issues existing in the current Jurion IPG system. Based on the results of an evaluation, we will be able to determine from a business point of view, which approach suits best for a real business use case.

6.1.6.2 Architecture

Figure 6.20 shows the architecture of the platform that was constructed to support the Jurion IPG use case. The IPG system is shown on the left. It consists of a CMS and an SQL database, upon which a suite of Business Intelligence services have been developed – some of which access the database directly and some of which use the API provided by the CMS. In this scenario, we compare two alternative approaches to solving the IPG problems.

Category	ID	Problem
Information is not there or is wrong	1	The lack of the trustee in bankruptcy proceedings
Information is not there or is wrong	2	Member of management board in companies where management board doesn't exist
Information is not there or is wrong	3	Member of management board without function in board e.g. board chairman, vice chairman
Information is not there or is wrong	4	Commercial proxy without type of proxy e.g. joint commercial representation
Information is not there or is wrong	5	Proxy without type of proxy
Information is not there or is wrong	6	Lack of additional information about way of appointment of a trustee
Information is not there or is wrong	7	Limited partner without limited liability amount
Information is not there or is wrong	8	Are there multiple shareholders if company is labeled as „Sole Shareholder“
Information is not there or is wrong	9	Information about the suspension of a member of the management board - only YES or NO
Information is wrong	10	The same person in management board and as commercial proxy
Information is wrong	11	Member of management board is a member of supervision or a commercial proxy, a official receiver, a trustee
Information is wrong	12	Receiver is a member of supervision or a trustee
Information is wrong	13	Official receiver is a member of management board or a member of supervision ,a trustee, an appointed person
Information is wrong	14	Trustee is a receiver or an official receiver, a member of management board
Information is wrong	15	There should be at least one person (natural or legal) in representation (management board, partners, trustee)or receiver / official receiver at any moment in time.
Information is not there or is wrong	16	Partners without information like amount of shares
Information is not there or is wrong	17	Do value of partners shares at every moment in time is equal or lower than capital value.
Information is not there or is wrong	18	Lack of amount of capital value in joint stock company and limited liability company
Information is not there or is wrong	19	Lack of information about way of formation of a company only information about circumstances of formation
Information is not there or is wrong	20	Lack of information about circumstances of formation of a company only information about way of formation
Information is not there or is wrong	21	Did a company publish multiple annual reports.
Information is not there or is wrong	22	Lack of the post office in company address when is not the same like the place where headquarter is
Information is not there or is wrong	23	Lack of a date of validation of expunging company from the court registry
Information is not there or is wrong	24	Lack of title of organ - supervisory board
Information is not there or is wrong	25	PESEL No. with less digits than 11 when first digit is 0
Information is not there or is wrong	26	Email and web page address with space
Information is not there or is wrong	27	Lack of @ in email address
Information is not there or is wrong	28	@ in webpage address
Information exists, but is hardly understandable	29	Complexity of the data model in the table describing attributes for company and relationship – the table szczegol_instyt_watroszc
Information exists, but I can't do anything with that – no process for consuming info	30	Information from the legal notice about a ban on economic activity and ban to be a member in representation (management board), supervision (supervisory board or audit committee)
Process exists, but isn't working or is too slow so solution is unknown	31	Find out relationship at specific moment in time between company and company, company and person, person and person
Information exists, but is difficult to get	32	Cycle loop - find out if company A is owner of company B, than company B is owner of company C where C is owner of company A

Figure 6.19 Jurion IPG unsolvable issues.

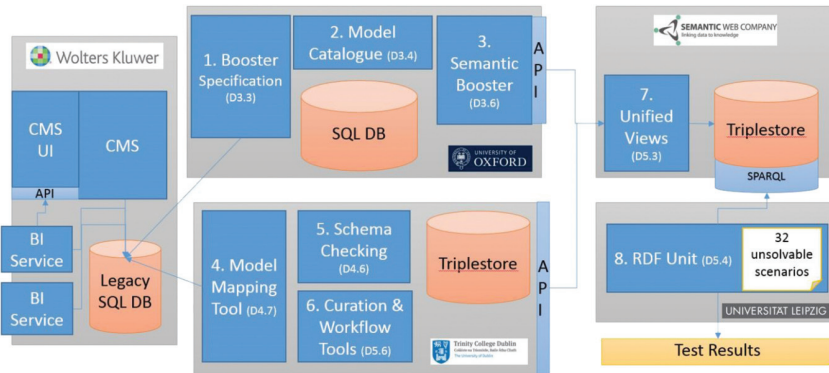


Figure 6.20 Jurion IPG use-case architecture showing integration across all major project tools and partners.

The first approach is provided by a configuration of the Oxford MDE approach. A booster specification is created (1) which generates SQL statements to extract the data from the legacy SQL DB and saves it in a format that can be managed by the Model Catalogue tool (2), the booster specification should ensure that these extracted data are correct by construction according to the booster specification. Then, these data are made available as RDF via Semantic Booster.

The second approach is provided by a configuration of Dacura services developed at Trinity College Dublin. The Model mapping tool (4) transforms the SQL schema of the legacy DB into an OWL ontology which is then used by the schema checking tool (5) to ensure that all data conform to the model. The curation and workflow tools (6) allow data managers to change the model and migrate the data and manage the process.

In order to properly compare the results of the two approaches, the RDF that they produce must be mapped to a common model – the UnifiedViews tool (7) provides this service and saves the resulting data to a triplestore. Finally, RDF Unit is used to test the output against the 32 unsolvable scenarios shown in Figure 6.19 to evaluate the success of the competing results. As RDFUnit supports arbitrary SPARQL queries, it is possible (although sometimes inconvenient) to encode all the evaluations as RDFUnit tests.

6.1.6.3 Tools and features

The tools and features used in the JURION IPG system are detailed in Figure 6.21.

Software	Type	RESTful API	Triple Store	SPARQL	Linked Data	Shared Ontologies supported
PoolParty	Platform	X	X	X	X	RDF(S), PROV, SKOS
Model Catalogue	Platform	X				RDF(S), OWL, PROV
RDFUnit	Command line tool		X	X	X	RDF(S), PROV, DQV, DataID, SHACL, RUT,
Semantic Booster	Command line Tool	X	X			RDF(S), OWL, PROV

Figure 6.21 Integration Paradigms and vocabularies supported by ALIGNED tools and platforms.

6.1.6.4 Implementation

Modifying Seshat Schema: Dacura provides tools to allow users to edit and modify ontologies on the fly (Figure 6.9). Using the Dacura browser plugin, users can browse the current Seshat code book and create properties and objects, adding them to the Seshat ontology and allowing researchers to collect information on these newly added properties.

Data complexity: Wolters Kluwer has managed and is still managing a tremendous business transformation process from a publishing house to a global information service provider (Figure 6.22). This development requires that high value-added services like IPG are also transformed from a traditional monolithic technical environment to a modular, flexible and sustainable infrastructure. Due to its data complexity and data quality issues (e.g., the main added value lies in the complex relationship model), ALIGNED tools can heavily support this transformation process.

Semantic Booster and the Model Catalogue. Semantic Booster has its strengths in the automatic model and software code creation process. It has also strong quality constraints so that no invalid data get into the transformation process. This approach was augmented by using RDFUnit for further data quality checks and which is the prerequisite to connect external open datasets to the IPG application in an easy and sustainable way (see Figure 6.23).

A booster specification is created which creates a model from the SQL database, along with formal constraints, which ensure that the data remain correct by construction. The Model Catalogue tool is then used to manage this data model. Semantic booster is used to make these data available as RDF via an API.

Data Complexity

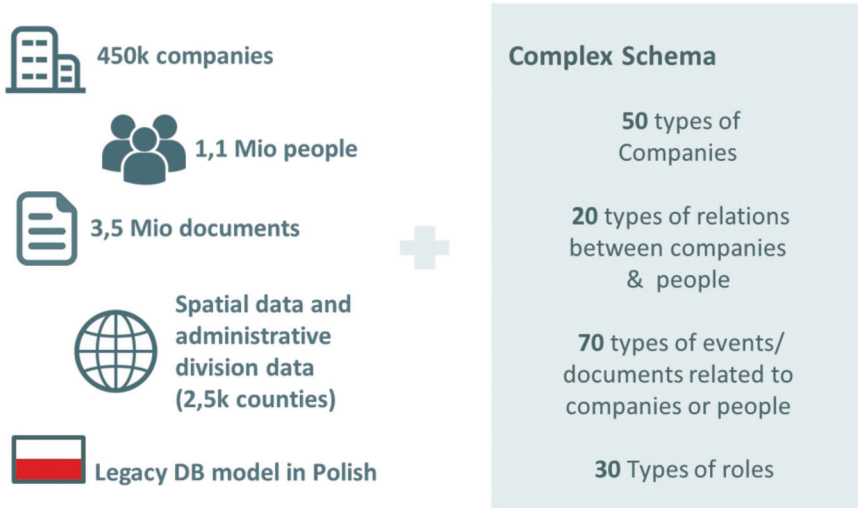


Figure 6.22 Complexity of the Jurion IPG use case.

How Aligned Tools are integrated



We use the **Model Catalogue** to collaboratively manage a subset of the IPG data model. Future Releases will broaden the scope of the model.

Semantic Booster is used to generate a working implementation of the model in a SQL database with API and GUI. This tool also enforces a range of data integrity constraints and business rules improving quality already.

In addition, Semantic Booster exposes the data as RDF triples which is then used by **RDF Unit** to run advanced data quality checks. Having the Data in RDF opens us further for linking and integration into existing semantic web tools and data sets.



Figure 6.23 Integrating Semantic Booster and the Model Catalogue.

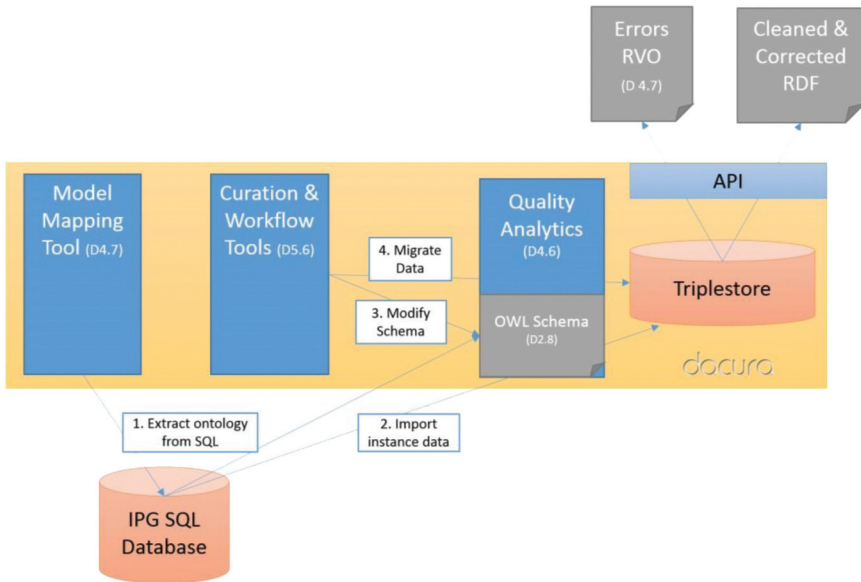


Figure 6.24 IPG Data Error detection and correction using Dacura.

Dacura provides an alternative method of achieving the same results (Figure 6.24). Firstly, the model mapper tool is used to generate an OWL ontology from the IPG SQL table structure (Figure 6.25). This ontology is deployed as the schema for the graph into which the instance data are imported.

Dacura's curation tools provide user interfaces which enable the data manager to view and modify the data and to analyse it for validation errors (Figure 6.26). The manager can use these tools to change the schema to include complex constraints on data quality. The results are provided as a cleaned, schema conformant RDF dataset and a list of errors expressed using ALIGNED's RVO ontology.

Unified Views: in order to ensure that the results of the validation processes carried out by Dacura and Semantic Booster can be evaluated, they must be mapped to comparable schema for testing. The Unified Views tool is used to manage this mapping and transformation and to save the transformed data to a triple store.

RDFUnit: Each of the unsolvable issues is encoded as RDFUnit scripts, which run SPARQL queries against the final data to check whether the issues are still present in the data. These queries are run against both the data

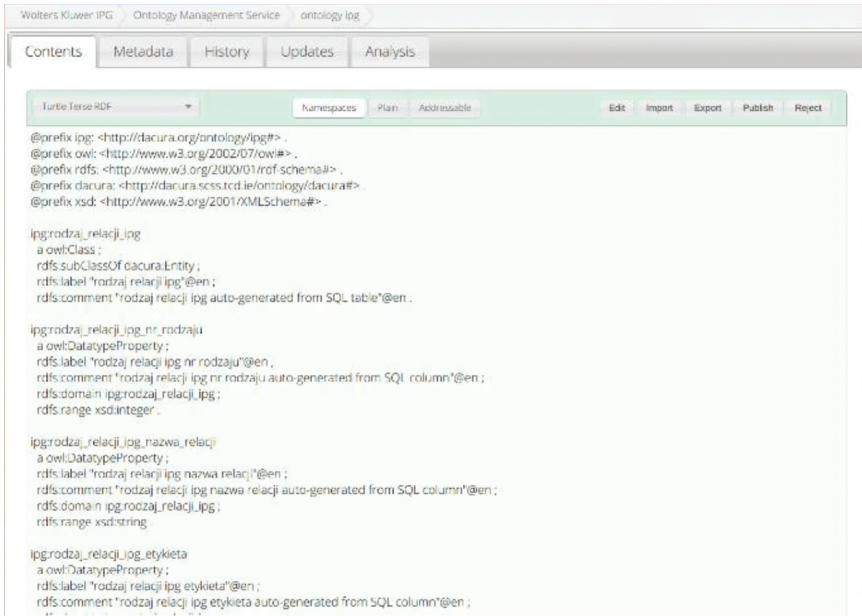


Figure 6.25 Ontology generated from IPG SQL database by Dacura’s Model Mapper Tool.

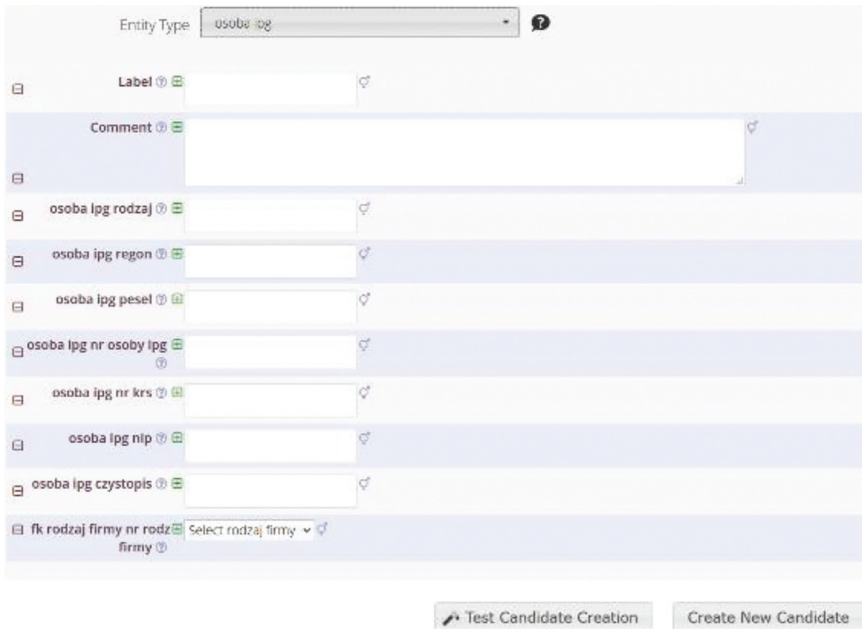


Figure 6.26 Using Dacura’s curation tools to analyse the IPG data model.

produced by Dacura and by Semantic Booster. In situations where issues still remain, RDFUnit can be used to fix some of these outstanding issues.

Conclusion: The platform produces a result set which describes the errors that have been found in the data which can be passed back to the DB administrators for correction.

6.1.6.5 Evaluation

Measuring Overall Value

The IPG database supports a variety of business services for customers which provide revenue to Wolters Kluwer – a very clear measure of the value provided by the system. The most important metrics in this case are: firstly, the curation cost of maintaining a given quality of service as measured by revenue; and secondly, the cost of improving the overall quality of service to provide more value and increase revenue, for example, by adding new features and new business services to the system. As the scale of the system – both in terms of the size of the database and the complexity of the services consuming the data – has increased, the curation costs have increased to such a stage that the cost of improving Quality of Service may be greater than the increase in business value that will accrue.

Data Quality Dimensions and Thresholds

Accuracy of certain information is very important in a commercial intelligence system. For example, accurate identification, accurate contact information and accurate shareholder and other relationship information are all significant in terms of the overall value provided to customers by the system. Users will tolerate some errors, but there is a threshold at which they will lose confidence in the value provided by the system.

Model Agility

The existing data model in the IPG SQL database have evolved to a stage where it is difficult to understand. Database access is heavily optimised and any changes to the structure require modification of caching and other optimisations. In addition, if we change any part of the structure of the database, we will likely break existing programs which use that part and it is very expensive to change the code of existing programs. Rather than attempting to modify the existing data model, a path to migrating to a new and easier to understand data model is required. In this case, the task is to

create a new data model that is easier to understand than the existing model without increasing the overall complexity of the system.

Data Agility

Once again, scale is the obstacle to overcome in enabling data agility in IPG. The overall system is highly optimised, and it requires significant effort to integrate any changes into existing infrastructure, testing, and so on, before we can safely ensure that the new service will not negatively affect the overall QoS of the system. Particularly problematic are queries that contain many complex joins and programs that make repeated round trips to the database, for example by executing a query in a loop, as they can put high load on the server and decrease the QoS across the entire system. To evaluate the overall system's data agility, the task is to create a new application, based on existing data, which identifies illegal relationships between people and companies.

Task 1: Curation

Identify and correct 10 different types of significant errors in the database.

Before ALIGNED

- Write queries to extract all the fields to be analysed from the database
- Write and maintain functions to analyse the extracted fields for the various types of errors identified and apply auto-correction where possible.
- Develop and maintain a system which allows users to view and correct identified errors
- Operate this system until the error rate has fallen below the desired threshold.
- Write queries to insert the corrected values back into the database.
- Either of:
 - Trace the source of errors back to the programs that produced them, fix and redeploy the programs
 - Periodically test the Database and rerun the process if quality levels have fallen below minimum thresholds.

Task 2: Model Agility

Create a new, easier to understand model and deploy it so that all new services can use the new model while existing services still use the old model, with

state being shared between them, without increasing the overall complexity of the system, increasing the data curation costs or reducing agility.

Before ALIGNED

What makes this problem hard is that, with current technologies (SQL), this requires a complete refactoring of the database and all the services that use it and the solution, at best, slightly improves the problem by slightly increasing the scale limit at which agility drops to zero. If we allow the existing data model to remain untouched, so that we do not have to change the code of existing services, and use middleware or an ORM architecture to create a new model for new data and link it to the existing model through code, we do not solve the problem at all – it actually makes it worse by introducing another layer of code that must be understood and maintained in order to understand or change the data or model. Thus, the only real alternative, using current technologies, is a complete refactoring of the database and re-engineering of the entire system and all of its services to use the new database schema.

Carry out complete system refactoring, with new, simplified schema, existing data mapped to that schema and all software services updated to use new model.

Task 3: Data Agility

Implement a new application which uses the IPG data to identify several different types of ‘illegal’ relationships between people and companies.

Before ALIGNED

- Write, test and maintain complex recursive SQL queries using CTE or CONNECT BY syntax to identify relevant instances.
- Write and maintain program to execute queries against database and return results
- Write and maintain program to browse and display result

6.1.6.6 Experimental evaluation

We used ALIGNED tools to address 31 unsolvable problems identified by IPG. This required us to complete the following tasks:

- Create a semantic model to represent the entities and relationships referenced in the IPG unsolvable problems.
- Create a mapping from the IPG SQL schema to the ontology

- Import the data to the ontological version and analyse it using the 22 quality constraints represented statically in the ontology
- Run a set of queries against the knowledge base to identify 8 of the 31 problems that could be expressed as graph queries.

We measured the time and effort required to complete these steps and identify all constraint violations in the SQL data, covering 30 of the 31 problems. The vast bulk of the effort was expended on scaling our reasoner and storage engine and toolchains to handle the scale of the data, as, when transformed into semantic representations, the IPG database amounted to tens of billions of triples.

We established a pipeline which lifts IPG’s SQL schema to an OWL ontology and then converts SQL row data into triples. We then transform the data triples using our Dacura mapping tool into an ideal schema. At this point, the instance data are checked for consistency with the ontology.

The results of the consistency check are reported as a file of JSON objects which elaborate the problem and its source. After running the consistency check, we found an initial set of 2,103,583 errors. A high proportion of a sample of these errors are genuine and have been verified (>95%), but since work on the project is ongoing, the number of false positives has not been completely determined.

The ontology design phase took around two days of expert ontology designer time. Since most of the importation is automatic, the process took around one day of developer time. Mapping of triples into the ideal schema took an additional one day. The total development time is then around one working week of effort.

Since IPG’s original use case described “unsolvable” problems, i.e., problems deemed too difficult to solve given the state of the current software and database setup, setting a baseline is somewhat difficult.

However, these problems are not genuinely unsolvable in abstract but simply too time intensive and expensive to solve. We can estimate the time that it would take in excess of one month of developer time, and very likely 6 months in order to find the ~ 2 million errors. This gives a range of speedups from using our methodology of between 400% and 2400%.

6.2 Seshat – Collecting and Curating High-Value Datasets with the Dacura Platform

The Seshat: Global History Databank is an international initiative of humanities and social science scholars to build an open repository of expert-curated

historical time-series data. The Seshat project began by selecting a sample of 30 areas from around the world. For each area, they recorded all societies that had controlled it throughout history, and answered over a thousand questions about each – describing its population, technology, religion, infrastructure, and so on. This made it possible to answer a wide range of questions about each of them – describing its population, technology, religion, infrastructure, and so on. The Seshat has been designed to test theories about the evolution of social complexity, from the point of view of historians and anthropologists. The databank extracts data from a combination of databases, Linked Data, websites, academic publications, and human experts.

A special code book defined the full list of questions, and researchers added data to the system by creating a copy of the code book page for each society, and adding data points using a special syntax that encoded uncertainty, disagreement, and temporal scope, along with comments and citations in relation to domain-specific provenance information. In the initial stages of the Seshat project, a wiki was used to collect the data. The system amassed over 200,000 data points on hundreds of civilisations, but whilst the unstructured wiki data store allowed great flexibility at the start of the project, it did not scale to the number of contributors, data users, data points, or the complexity of the data.

The Seshat evolved to encompass new areas that were not originally anticipated. In particular, this involved recording societies from the prehistoric past, which required a collection of archaeological data. It soon became obvious that many Seshat variables were unsuitable for capturing this part of human past. There was also a lack of relevant proxies that would allow translation of archaeological evidence into coding templates. Accordingly, the Archaeological Seshat code book was designed and developed in order to fill in the gap, and the data were collected independently.

A wiki-based approach, used in Seshat for the data collection task, posed numerous problems, in particular for the verification of data correctness, and the extraction of data in usable forms. As the dataset grew and the focus moved from collection to integration and analysis, several other significant problems emerged. The fundamental problem is that a wiki is designed for human presentation and editing of data. To a machine, it is semi-structured, which lacks any type information and the meaning of the elements depends on their context within a jumble of HTML. Without any support for validation, errors proliferated.

The limitations of the wiki also impacted agility. As the Seshat code book was rapidly evolving, any changes needed to be manually copied to

all existing data pages was a costly and error-prone task. There was also no easy way to express spatial data through the wiki, so these data were stored in separate GISs. This solution also offered no support for publication, while the scraping tool could extract the raw datapoints, citations, and comments were also important but were encoded in totally unstructured HTML.

Productivity suffered as increasing resources had to be devoted to curation and cleaning. Some of the corrections were not copied back to the wiki, and spreadsheets became the authoritative source for some sections of the data. Moreover, there was no way of incorporating third-party data into Seshat dataset.

6.2.1 Use Case

6.2.1.1 Problem statement

A group of researchers, distributed geographically and across multiple teams and disciplines are collaborating on the compilation of the Seshat dataset describing human social evolution since Neolithic times. The goal is to record geo-temporal time-series datasets describing how hundreds of variables describing social complexity changed with time and place. The Seshat researchers are currently using a wiki and a polity-based template, which includes a simple syntax for encoding machine-readable variable values, to collect the data. The Seshat researchers can currently be roughly broken down into three roles: harvesters – typically RAs who are paid to input data to fill the datasets on a per-polity basis; experts – scholars with specific expertise in particular geo-temporal slices of human history, their role is to correct, interpret and validate the data for their particular areas of expertise; architects – the core Seshat editorial team, who are responsible for designing and modifying the dataset Schema. The high-level goal of ALIGNED in this use case is to produce tools for the Seshat researchers which will increase productivity and data quality and improve the availability of data for analysis.

Actors

<i>Role</i>	Description
<i>Harvester</i>	non-technical users who add and update data
<i>Editor</i>	moderate, correct and manage the data in the system over time
<i>Expert</i>	use domain-specific knowledge to analyse and interlink data in the system
<i>Architect</i>	make changes to the schema and manage transitions between schema versions

The requirements on which the Seshat use case was based are detailed in Appendix A.

6.2.2 Architecture

Figure 6.27 shows the architecture of the system that was developed to support the Seshat use case trials, highlighting the places where common ALIGNED integration paradigms and ontologies were exploited. The system demonstrates integration between three of the project’s major suites of tools, and three of the project’s use cases.

The full suite of Dacura tools form the core of the system, importing the data, ensuring it meets consistency requirements, automatically producing user-interfaces and curation tools to allow the expert contributors to use the system without any knowledge of the underlying semantic technologies being used, and finally publishing the data and making it available to software engineers.

The Model Catalogue tool was used to help develop and manage the ontologies used by the system – it supports OWL models and provides a RESTful API to support easy integration with third-party tools and incorporating into complex workflows.

The Unified Views tool, developed as part of the SWC’s PoolParty semantic suite was used to manage the integration of datasets from third-party datasets into the Seshat dataset. In this case, the DBpedia knowledge base was the data source being exploited.

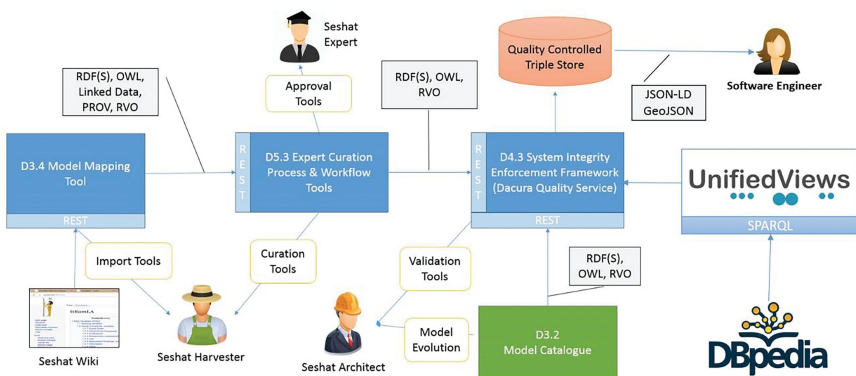


Figure 6.27 Seshat Use Case Trial System Architecture, showing the tools provided to different Seshat users, the use of ALIGNED integration standards and interoperation paradigms.

The final platform thus directly incorporated the research outputs of six of the seven research groups involved in ALIGNED and demonstrated integration across three of the use cases – DBpedia provided data and the Pool-Party use case provided tools and expertise in establishing the data import pipelines.

In April 2017, this platform was used to prepare and publish the first public release of data from Seshat, which in an attractive and well-structured format for appraisal by other researchers – particularly focussed on scientific reviewers who needed to evaluate the data on which several of the major Seshat publications were based. Since then, the project’s major focus has been made to deploy the system in a software engineering context, which has involved making the RDF/OWL data stored within the system available in simpler forms, such as GeoJSON and JSON-LD available to software engineers.

The platform has been constructed to support the following Seshat data curation tasks:

- Importing the large volume of wiki data that they have accumulated in a semi-structured form, into a structured, rich semantic format according to a pre-defined model, that is amenable to statistical analysis and automated quality control.
- Analysing the data to identify a large number of new constraint violations – e.g., datatype constraints, referential integrity constraints, cardinality constraints. In the current Seshat data collection workflow, such problems only show up at data-analysis stage and it requires a very significant manual effort to amend them at that late stage.
- Providing approval pipelines and workflow tools to allow moderators to inspect and correct problems identified in the data and to give them the agility to be able to use lower-skilled data collectors with higher error rates without sacrificing overall quality.
- Providing model rapid prototyping tools to allow our archaeologist partners to experiment with the definition of large new segments of the Seshat schema to allow them to define semantic mappings between entities at different levels of abstraction and time-depth. This supported the accumulation of archaeological evidence and extended the time-depth of the Seshat ontology which was initially conceived primarily to investigate societies that were historically known.
- Importing data from third-party datasets such as DBpedia and Pleiades historical gazette and integrating it with the existing Seshat data.

Software	Feature	Used for
Model Catalogue	Model Definition User Interfaces	Model Prototyping
Model Catalogue	Model Export to OWL	Model Integrity Enforcement
Dacura	Model Mapping Tool	Importing wiki data
Dacura	Real-time Instance Data Validation	Testing imported data
Dacura	Model Generated User Interfaces	Correcting imported data
Dacura	Curation Workflow Tools	Update approval queues
PoolParty	Unified Views ETL	Import third party data

Figure 6.28 Features of the ALIGNED tools used to support the Seshat trials.

6.2.2.1 Tools and features

Figure 6.28 shows which features of the ALIGNED software tools have been deployed in order to support these scenarios.

6.2.3 Implementation

6.2.3.1 Dacura data curation platform

The Dacura Linked Data curation platform¹ is developed at Trinity College Dublin. Dacura provides support for dataset capture, curation, and publication. The major components of Dacura in the context of the Seshat use case are shown in Figure 6.29.

In the initial prototype developed for Seshat four of the components from Figure 6.29 are used: (1) the wiki data entry/validation tools (top left in the figure) which are user-facing data curation widgets; (2) the schema management tools which include the Model Catalogue tool in the demo; (3) the data quality controls (lower middle of the figure) which perform schema and data integrity checks and act as a data quality gatekeeper for the RDF triple store; and (4) the data export tool or wiki scraper which can transform Seshat data into the TSV file dumps required by statistical analysts within Seshat.

6.2.3.2 General description

Dacura provides tool support to improve the efficiency and accuracy of Seshat's data collection processes.

- The wiki data entry/validation tools make data entry easier for Seshat researchers. This also assists in collecting more complex data and data validation at the point of entry.
- The schema management tools check that OWL-based schemas are consistent and correct as they grow.

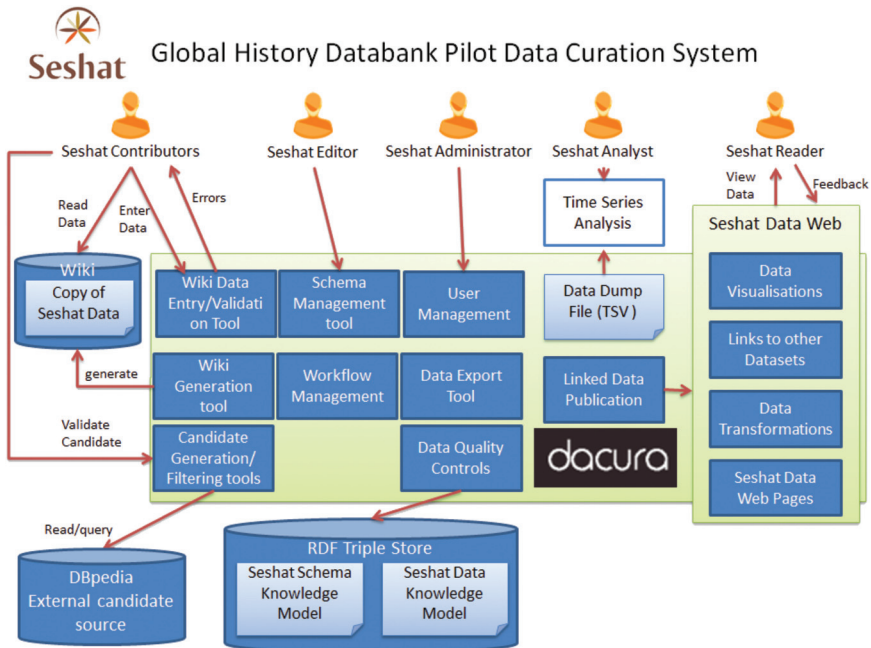


Figure 6.29 The Dacura platform in the context of the ALIGNED Seshat use case.

- The data quality controls ensure that both data entered through widgets and data already entered in the Seshat wiki is checked for conformity with the Seshat schema before it is added to the triplestore.
- The data export tool allows multi-format data publication. It also allows Seshat administrators to get a first look at how their dataset is growing and evolving.
- The Seshat OWL ontology developed for this demonstrator and used by our tools enables more structured information to be captured than the original Seshat.

6.2.3.3 Detailed process

This section describes the use of each of the components developed for the demonstrator system, along with a screenshot of the components in use where applicable.

Dacura data entry validation tools (Figure 6.30) are embedded in the Seshat wiki, allowing researchers to validate previously entered data and add new variables to the dataset. Researchers can validate or enter data directly

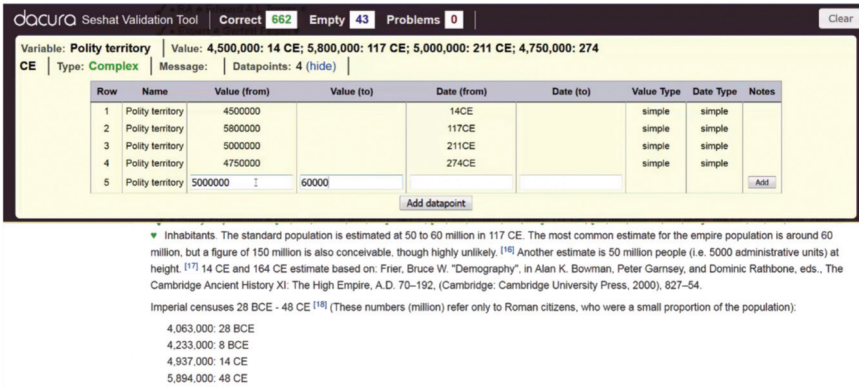


Figure 6.30 Screenshot of TCD’s Seshat Data Entry/Validation tool in Demonstrator System.

from the wiki page. These tools reduce the complexity of entering data in the wiki, as the need for complicated syntax is reduced and any errors in data will be immediately revealed. A version of this tool that supports validation of data entered into the wiki has already been deployed in the live Seshat system.

Modifying Seshat Schema: Dacura provides tools to allow users to edit and modify ontologies on the fly (Figure 6.31). Using the Dacura browser plugin, users can browse the current Seshat code book and create properties

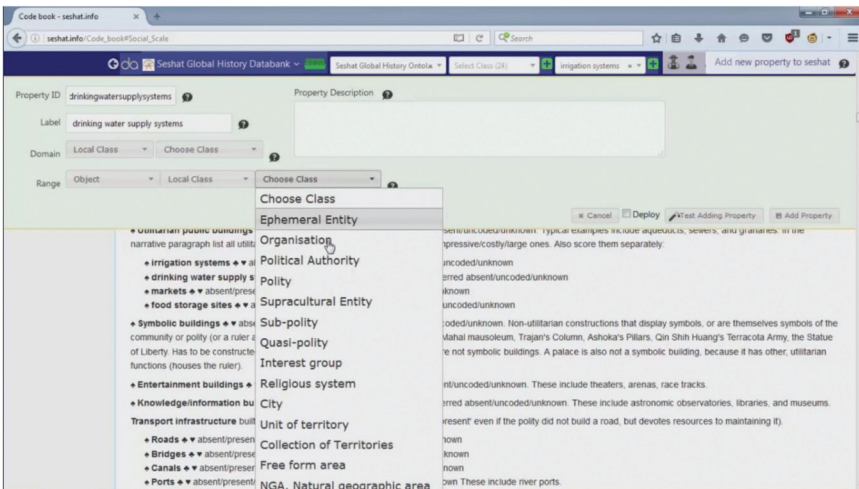


Figure 6.31 Modifying Seshat Schema.

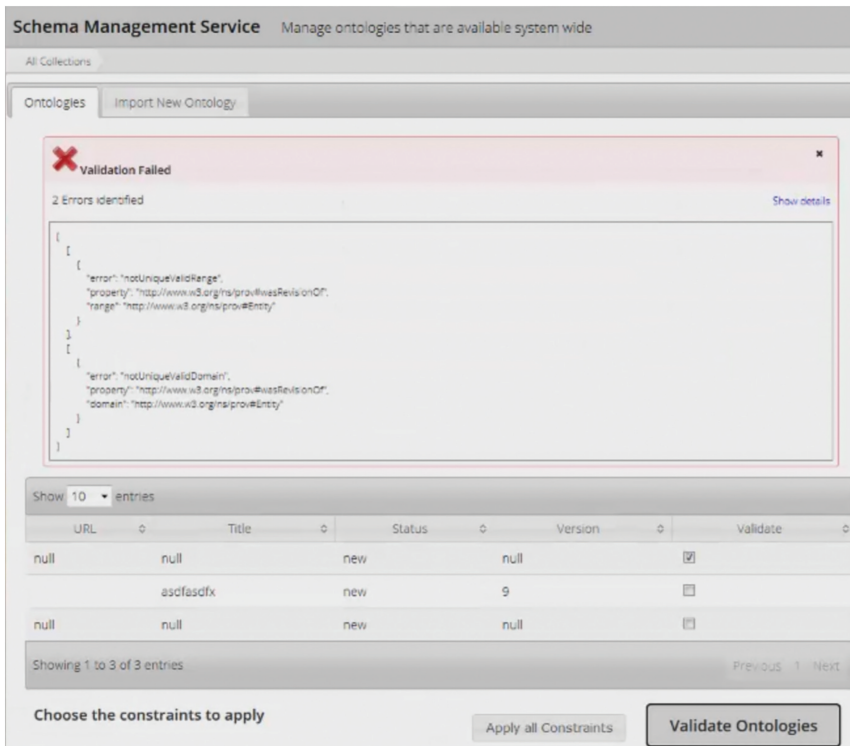


Figure 6.32 Screenshot of TCD’s Schema Management component using the prototype integrity enforcement framework in the Demonstrator System.

and objects, adding them to the Seshat ontology and allowing researchers to collect information on these newly added properties.

Triplestore integrity enforcement (Figure 6.32) is a key feature of Dacura. Preventing data that are not in accordance with the schema or preventing a malformed schema from entering the triplestore ensures that all data are of high quality. This reduces the need for Seshat researchers to spend time correcting errors in the dataset. Dacura checks that imported vocabularies are consistent before allowing them to enter the triple store and constrain instance data.

The schema management components (Figure 6.33) in Dacura allow changes to the schema to be analysed to ensure that the schema remains consistent. Checks are performed on schemas before they are used in the data store, highlighting errors and potential issues for attention. A range of

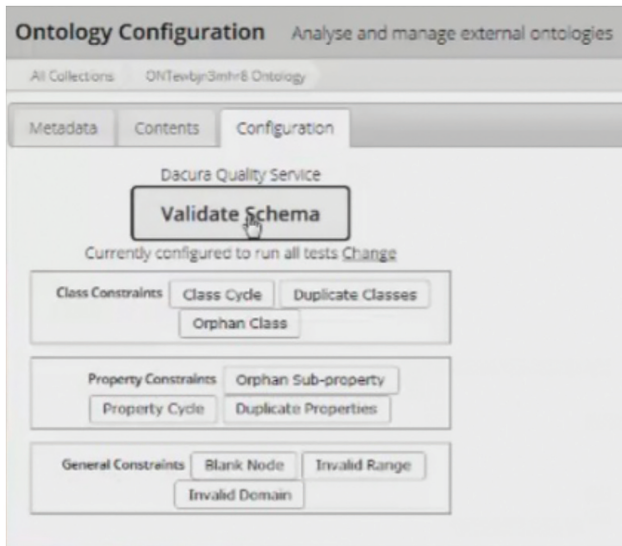


Figure 6.33 Screenshot of TCD's Schema Validation Service in Demonstrator System.

checks are performed using a constraint-based interpretation of the Seshat OWL ontology.

Finally, the wiki export component (Figure 6.34) extracts the historical data entered in the Seshat wiki, parses them, and produces a TSV of these values ordered by date. This allows Seshat administrators to perform analyses without needing to manually extract the values from the large and constantly growing dataset. It also produces error reporting, allowing researchers to identify errors in the dataset and see how the data are evolving.

The introduction of the Dacura data validation component into the live Seshat data collection process has reduced the rate of errors in the Seshat wiki. Despite a large increase in the size of the wiki of 29% (from 56,160 to 72,252 data points) between March and June 2015, the absolute number of errors has decreased by 19%. The rate of errors per variable has decreased by 42%, from 0.035 errors per variable to 0.02 errors per variable. This shows the positive impact of deploying data quality/data curation tools on the Seshat workflows. This trend is shown in Figure 6.35.

Managing complex workflows: The Dacura approval queue allows dataset administrators to monitor added data for quality and completeness (Figure 6.36). Administrators can approve, deny, publish and unpublish the Linked Data objects submitted by Seshat researchers.

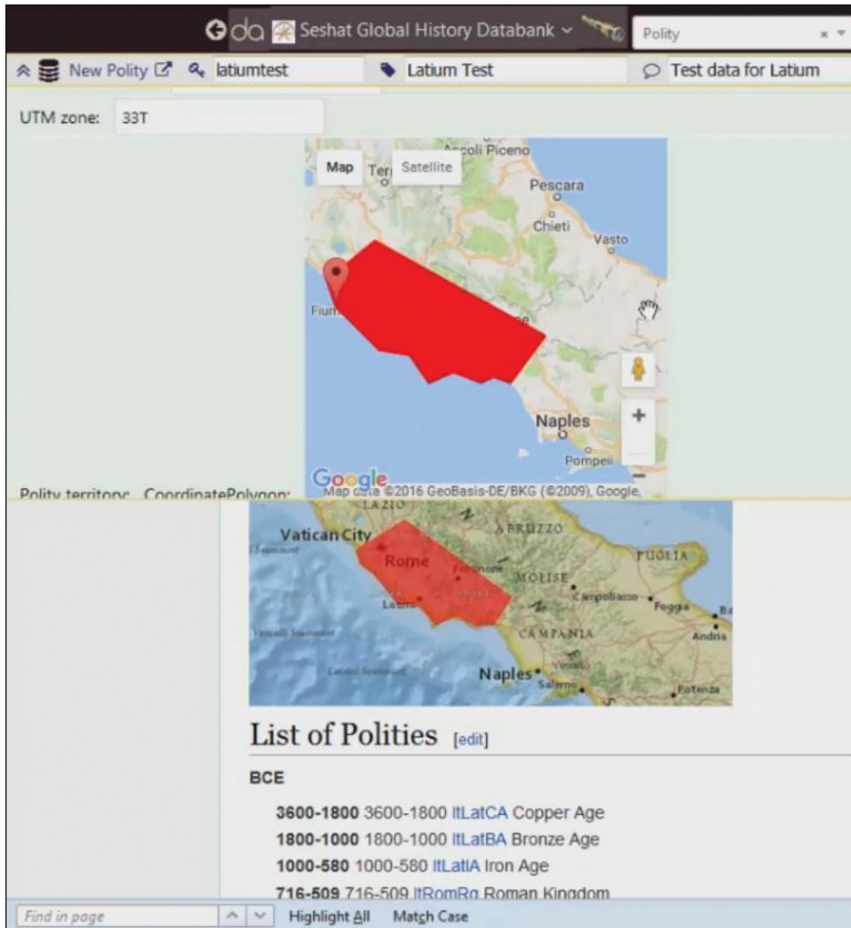


Figure 6.34 Screenshot of TCD's Wiki Export Component.

Importing third-party datasets: The Unified Views tool (Figure 6.37) allows data to be imported via SPARQL from third-party datasets, in this case, DBpedia is used as a source of data. Unified Views allows the establishment of processing workflows to automate the importation of such data.

Publication: The Dacura system allows the generation and publication of processed and curated data in easy-to-use forms. Casual users can browse Web pages (Figure 6.38 which lay out the information in a simple and structured manner). Seshat team members looking to perform analysis on

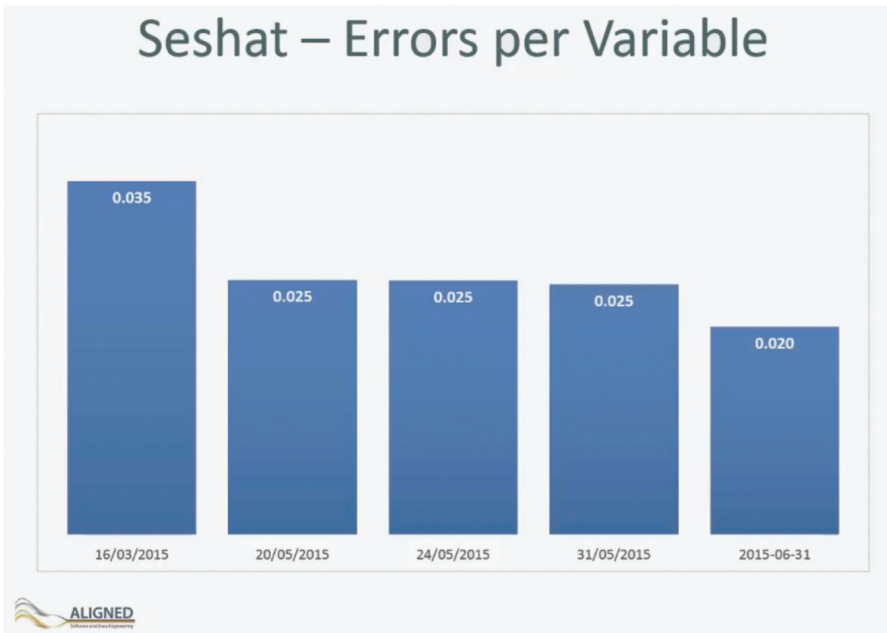


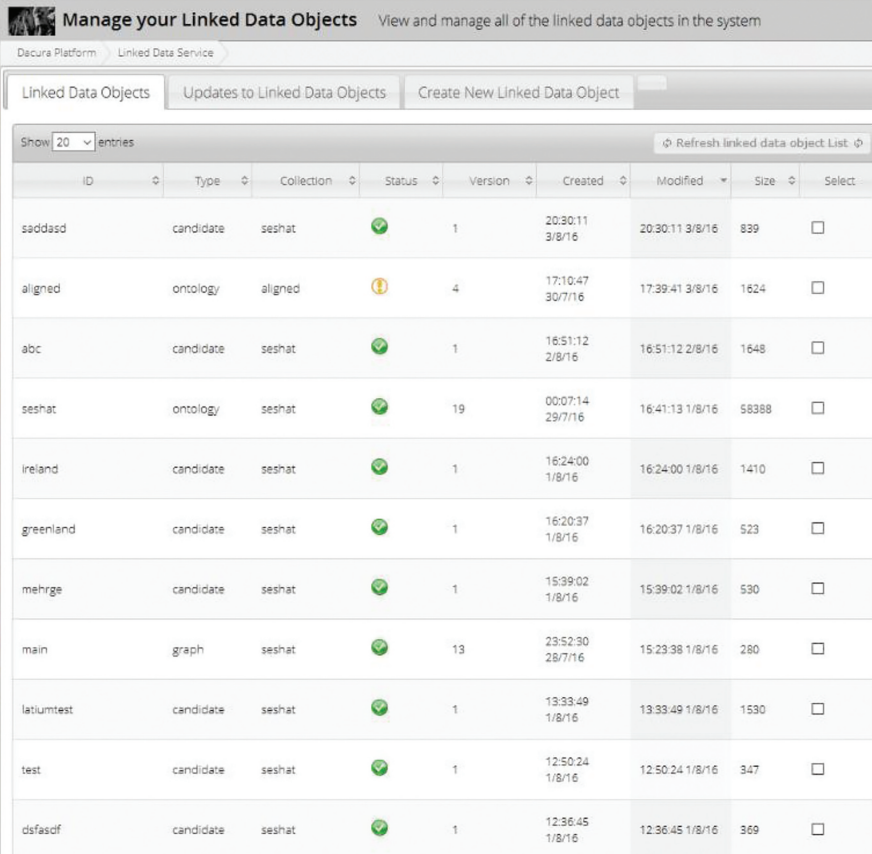
Figure 6.35 Seshat Errors per variable.

the data can access it in structured forms, which can be easily imported into analysis software.

Software Engineering support: The Dacura system provides several services to software engineers developing software that utilises the data curated by the system (Figure 6.39). These include reliable access to version controlled data models at a well-known URL, change notifications and the automatic production of simpler formats which are more familiar to traditional Web-developers. In this case, a GeoJSON stream is automatically made available describing all the features in the dataset that have a geographical location associated with them.

6.2.4 Overview of the Model Catalogue

When dealing with large, complex datasets, it is important to have tools to help collaborators understand what each data point means, how it has been collected, and how groups of data points may be interrelated. Typically, a large number of tools are used for this kind of metadata management:



The screenshot shows a web interface titled "Manage your Linked Data Objects" with the subtitle "View and manage all of the linked data objects in the system". Below the title are navigation tabs: "Linked Data Objects", "Updates to Linked Data Objects", and "Create New Linked Data Object". A "Show 20 entries" dropdown and a "Refresh linked data object List" button are also visible.

ID	Type	Collection	Status	Version	Created	Modified	Size	Select
saddasd	candidate	seshat	✓	1	20:30:11 3/8/16	20:30:11 3/8/16	839	<input type="checkbox"/>
aligned	ontology	aligned	⚠	4	17:10:47 30/7/16	17:39:41 3/8/16	1624	<input type="checkbox"/>
abc	candidate	seshat	✓	1	16:51:12 2/8/16	16:51:12 2/8/16	1648	<input type="checkbox"/>
seshat	ontology	seshat	✓	19	00:07:14 29/7/16	16:41:13 1/8/16	58388	<input type="checkbox"/>
Ireland	candidate	seshat	✓	1	16:24:00 1/8/16	16:24:00 1/8/16	1410	<input type="checkbox"/>
greenland	candidate	seshat	✓	1	16:20:37 1/8/16	16:20:37 1/8/16	523	<input type="checkbox"/>
mehrge	candidate	seshat	✓	1	15:39:02 1/8/16	15:39:02 1/8/16	530	<input type="checkbox"/>
main	graph	seshat	✓	13	23:52:30 28/7/16	15:23:38 1/8/16	280	<input type="checkbox"/>
latiumtest	candidate	seshat	✓	1	13:33:49 1/8/16	13:33:49 1/8/16	1530	<input type="checkbox"/>
test	candidate	seshat	✓	1	12:50:24 1/8/16	12:50:24 1/8/16	347	<input type="checkbox"/>
disfasof	candidate	seshat	✓	1	12:36:45 1/8/16	12:36:45 1/8/16	369	<input type="checkbox"/>

Figure 6.36 Managing Complex Workflows.

data dictionaries for storing information about variables and allowed values; data manuals or procedures for describing the intended meaning of data points; specifications of forms describing how data are to be collected; or diagrams describing relationships between groups of data points. The Model Catalogue toolkit is being developed at Oxford University for the purposes of collaborative editing and sharing of such documents within a common framework. Based on previous work on international standards for metadata registration and previously explored in the context of clinical research, the tool is now being developed and extended to support the Seshat use case.

The tools have been built with a model-driven software development process in mind: programmatic interfaces allow communication between the

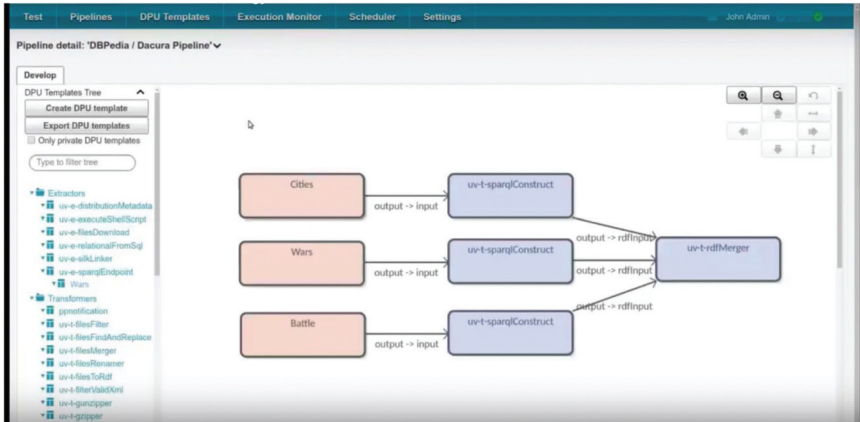


Figure 6.37 Importing data to Seshat from DBpedia with Unified Views.

Cambodian Basin



Figure 6.38 Publication.

catalogue and other systems; a number of export tools have been written to automatically produce or configure software artefacts such as databases or data messaging schemas from data model descriptions stored in the catalogue.

The extended tool is initially designed to support two key use cases – for the Seshat editors and data managers to cooperate in the incremental evolution and description of the Seshat data model or code book; and for researchers interested in using the Seshat data to understand which data points have been collected and their meaning and provenance. To support

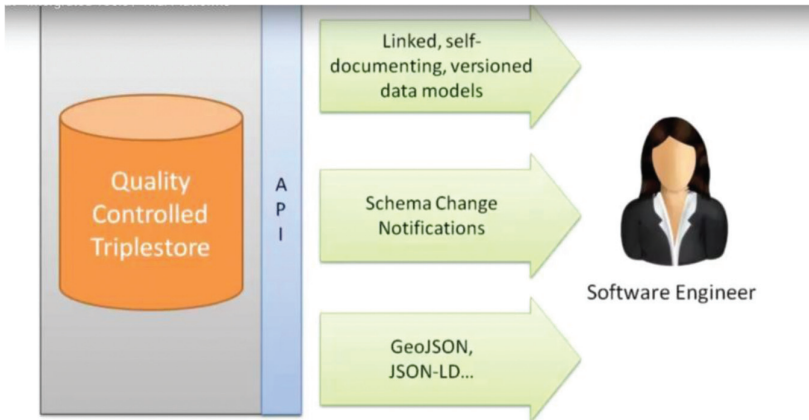


Figure 6.39 Services to support software engineering.

Draft Models

- Seshat Code Book 1.0
 - Main Variables (poly-based) 1.0
 - General variables 1.0
 - Social Complexity variables 1.0
 - Warfare variables 1.0
 - Military Technologies 1.0
 - Military Organization 1.0
 - General Characteristics of Warfare 1.0
 - List of Wars 1.0
 - Warfare coding template 1.0
 - General 1.0
 - War types 1.0
 - Cultural distance 1.0
 - Sieges, Battles, and Naval Engagements 1.0
 - Consequences for territories 1.0
 - Battle templates 1.0
 - Siege Template 1.0
 - Ritual variables 1.0

Cultural distance

Name	Description	Value Domain	Metadata
1 Same Ethnic Group	Both sides come from the same ethnic group and speak the same dialect of the common language. Most civil wars will fall into this category, unless there is a significant cultural distance between the two parties.	Absent Present Inferred present Inferred absent Uncoded Unknown	
1 Different dialects	Opponent groups perceive each other as having distinct ethnic identities demarcated by a significant dialectal difference, but still speaking a mutually comprehensible language but within the same linguistic group.	Absent Present Inferred present Inferred absent Uncoded Unknown	
1 Different languages	The opponent groups speak mutually unintelligible languages, but similarities between languages are apparent to them. Examples include English and Dutch, or Italian and Spanish, but not English and Italian (although both are Indo-European languages, the kinship between them is not apparent to a nonspecialist).	Absent Present Inferred present Inferred absent Uncoded Unknown	
1 Different language families	The opponent groups speak mutually unintelligible languages that have no similarities apparent to them. Examples include English and Chinese, or Italian and Hindi/Urdu (in the second example both are Indo-European languages, the kinship between them is not apparent to a nonspecialist).	Absent Present Inferred present Inferred absent Uncoded Unknown	
1 Different religious sects	This refers to different religious denominations with the same world religion. E.g., Catholics vs. Protestants, or Shites vs. Sunnis.	Absent Present Inferred present Inferred absent Uncoded Unknown	
1 Different world religions	Taoism, Confucianism (China), Hinduism, Buddhism, Jainism, Sikhism (India), Zoroastrianism, Judaism, Christianity, Islam (Middle East)	Absent Present Inferred present Inferred absent Uncoded	

Figure 6.40 The Model Catalogue user interface showing a section of the code book.

the first use case, the toolkit provides facilities for automatic import of existing documents and software artefacts to initialise data models, and uses careful structuring to minimise the amount of user input required. The Web-based editing environment (see Figure 6.40) promotes collaborative editing of modes, with processes for publication and versioning. To support the second use case, the Web interface allows exploration of data models, and the creation of user-friendly reports or exports. Data points can

be linked to provide additional meaning or context and can be compared to understand differences between different datasets, or a single dataset over time.

6.2.4.1 Model catalogue in the demonstrator system

This section describes the Model Catalogue components deployed in the demonstrator system. The section starts with a discussion of the Seshat use cases addressed by the components and then provides a description of the model curation processes supported by the Model Catalogue component. Finally, there is a subsection presenting some initial results from the deployment of the components.

General description

The Seshat databank is a complex dataset comprising more than 1,000 variables categorised into approximately 100 groups or classes. A code book describes each group and variable: a description about the intended meaning, or semantics, of the data points collected against it; a selection of possible values that the data point may take; links to related or similar variables in another category.

Furthermore, this set of variables has been evolving and expanding; in 3 years, more than 300 revisions to the code book have been made: variables have been added, removed, or extended; descriptions have been enhanced; the permitted range of values may have items added or removed.

This code book holds the key to understanding how data points should be collected and stored and how potential users of the data can make sense of what is made available. Until this point, the Seshat code book has been encoded into a wiki page. This has served the purpose required, but in order to scale, a new approach may be required. The Model Catalogue built by OxSE is intended to support:

- Collaborative editing of the code book as a curated data model
- Annotating variables and linking to related datasets or standards
- Versioning and publication life cycles for change management
- Informed reuse of data collected, through discovery and exploration and comparison of data models and variables
- Automated import and export of models into other data formats
- Generation or configuration of software artefacts within an iterative development approach

The Model Catalogue prototype makes progress towards schema evolution. The “Expert” and “Architect” user roles can modify the schema, allowing view-only access to the schema for “Harvester” and “Editor” user roles. There is also some progress towards addressing expert interpretation, where experts can use the Model Catalogue to understand and modify the code book to improve support for capturing complexity in the Seshat databank.

Detailed Process

The Model Catalogue stores concrete models such as dataset descriptions, form designs, database schemas, and so on, alongside an abstract representation common to all models. Each Data Model contains a number of Data Classes, which in turn may contain sub-classes and Data Elements, in our case representing the variables of the Seshat code book. This structure provides easy interoperability between models, and allows a simple tree-view for viewing and exploring models, such as in the screenshot shown in Figure 6.41. This view is useful for those exploring the structure of a data model: users interested in requesting or working with items of data, or editors wishing to make changes to the structure.

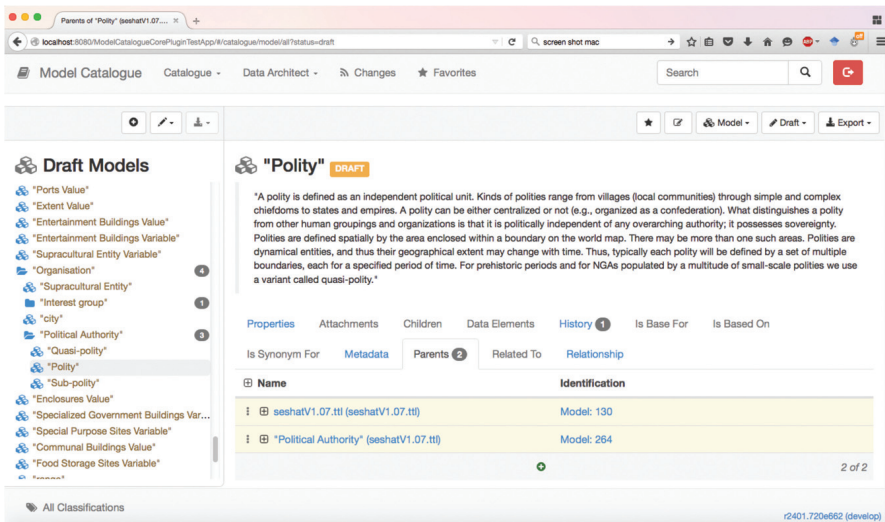


Figure 6.41 Screenshot of the Model Catalogue Web interface, showing the ‘tree view’ and a section of the Seshat code book.

Data models and their components may be linked with a number of different relationships describing the type of similarity between them. A plugin for discourse has been integrated: this allows users to comment on parts of a data model in a familiar fashion. Comments may include links to other data elements, or mention of other users, who can be prompted to respond. Attachments can be added too: links to websites, or file attachments giving more information about the meaning of a variable.

For prospective users of the data, it is important to understand how data collected against different versions of the code book may be related. Figure 6.42 shows a screen written for this purpose: highlighting differences in descriptions, sub-components (for models and classes), and datatypes (for data elements). Sophisticated search functionality allows data elements to be found within all models; all finalised models, or within the currently displayed model.

Model importers allow existing structures to be imported into the catalogue without manual transcription. One such importer has been written to ingest the OWL description of the Seshat code book, extracted by the TCD team for use in Dacura. Although written with Seshat in mind, this component may be re-configured for use in the other ALIGNED use cases during the next phase of the project.

Property / Name	seshatV1.08.ttl	seshatV1.07.ttl
Latest Version Id	275	275 129
Version Number	1	1
History	1 item(s)	1 item(s)
Is Based On	0 item(s)	0 item(s)
Classifications		
Is Base For	0 item(s)	0 item(s)
Has Attachment Of	0 item(s)	0 item(s)
Classifies	88 item(s)	88 item(s) 146 item(s)
	<ul style="list-style-type: none"> "battle" (seshatV1.08.ttl) "Building Variable" (seshatV1.08.ttl) "Building" (seshatV1.08.ttl) "City Variable" (seshatV1.08.ttl) "city" (seshatV1.08.ttl) 	<ul style="list-style-type: none"> "battle" (seshatV1.08.ttl) "Building Variable" (seshatV1.08.ttl) "Building" (seshatV1.08.ttl) "City Variable" (seshatV1.08.ttl) "city" (seshatV1.08.ttl)

Figure 6.42 Screenshot of the Model Catalogue Web interface showing the comparison between two versions of the Seshat code book.

6.2.5 Seshat Trial Platform Evaluation

6.2.5.1 Measuring overall value

Ultimately, the most important metric for the Seshat project is the cost of going from a hypothesis to a published scientific paper presenting an empirical evaluation of that hypothesis. Everything else is a means to that end. The nature of the variables chosen and the data collected are explicitly designed in order to enable computational analysis of particular hypotheses against historical evidence. The validity and significance of the analysis is then validated by the peer-review process of the world's top scientific journals [1, 2]. The best proxy for the value delivered by the overall system is the number of papers that are published in top-tier scientific journals.

6.2.5.2 Data quality dimensions and thresholds

seshat has unusually high Data Quality requirements across a large number of separate dimensions. The Seshat researchers need to be able to analyse the data statistically, which imposes high thresholds for syntactical accuracy and structural integrity. Furthermore, the historical accuracy of the data is also extremely important – as the Seshat researchers want to be able to identify patterns in long term historical processes in order to make predictions – these predictions can only be as good as the accuracy of their data. Because it is often the case that historical facts can only be known probabilistically, uncertainty must be incorporated into the data in such a way that statistical analysis can still be applied. Furthermore, it is the norm in top-tier scientific journals that datasets are scrutinised closely by reviewers. Because Seshat is pioneering new data-driven methods in the social sciences, it is thus particularly important that published datasets are robust in the face of expert scrutiny, as any significant errors risk undermining the credibility of the approach and not just the individual publication.

In terms of data agility, the most important requirements for Seshat are the ability to make data, of the required quality, available for analysis with the programs used by Seshat's data-analysts (R, Mathematica) and to make the data available for inspection for academic reviewers in whatever format will make the best impression upon them.

In terms of model agility, the most important requirements for Seshat is the ability to make changes to the 'code book' to reflect the input of new experts and experience. The Seshat code book has changed many times over the last few years as the community has grown and more expert opinion has been incorporated into the selection of variables and proxies to collect.

The Seshat data collection effort started some 2 years before the start of ALIGNED and has continued for almost 5 years at this stage. There are currently two major papers which are under review for publication in major top-tier scientific journals (both resubmitted with changes, as advised by the editorial committees). All the data that went into these publications were collected, curated, cleaned, and analysed using methods that pre-date the introduction of ALIGNED technology, with one exception – the publication of data for the reviewers. This provides us with a very good baseline measure with which we can evaluate the impact of ALIGNED technology: the total cost of producing these two papers – including all the cost that went into producing the final publication and associated datasets.

We can break it down into the following tasks:

1. Running expert workshops to identify interesting hypotheses and suitable proxy variables to form the dataset's schema.
2. Employing and training research assistants to fill in data on a wiki for each historical society of interest
3. Soliciting reviews from volunteer experts to validate the data entered in 2.
4. Developing and maintaining programs to extract data from the wiki, identify syntactical errors, make it available as a CSV for analysis and then transform it into the format required for each analysis and then, finally, perform the analysis and produce the results.
5. Employing and training research assistants to correct data entered in 2 in response to errors identified in 4, and reviews received in 3, to ensure that it remained accurate and true to the schema over time, as the schema changed to reflect the outputs of 1.
6. Developing and maintaining a program for publishing the data for appraisal by the general public and reviewers.

All these tasks were necessary for the production and publication of the journal papers and their associated datasets and must be included in the consideration of the system's productivity before the introduction of ALIGNED tools and methods. Furthermore, by looking at the cost of achieving required data quality levels as the schema was changed over this period and the cost of making data available in new ways, we can gain a reasonable estimate of the likely future productivity of the system. In order to demonstrate the impact of our tools and methods, therefore, we would ideally repeat the process from start to finish with our tools, starting from a new set of hypotheses and compare the overall cost of the two.

In December 2018, at a Seshat workshop in Oxford, in association with the ALIGNED general meeting, we began such an evaluation. In this study, we apply our tools and methods to the entirety of this process – from modelling tools to support 1, curation interfaces to support 2 and 3, data manipulation tools to support 4 and 6, error detection and correction tools to support 5. We have identified a set of new hypotheses that we wish to test and we will measure the total cost of transforming these hypotheses into published scientific papers. However, this is an irreducibly time-consuming task due to the nature of the domain. In particular, experts are a scarce resource and it inevitably takes considerable time to elicit all the domain knowledge necessary from the relevant experts for any particular historical question. Therefore, the full results of this comparison will not be known until the entire process is complete, which will be beyond the timeframe of ALIGNED.

However, what we can do in a short space of time is to measure the impact of our innovations on steps 4, 5, and 6 of the process and compare it with the existing system. The Seshat wiki contains several sections of data that have been collected by RAs and approved by experts but have not yet been cleaned, analysed or published. We choose one of these sections, comparable in terms of size and complexity to the datasets that were used to publish the first two papers. Starting from this point, we extract, clean, analyse and publish the data to the same level of quality as was achieved with the already published datasets and we measure the total cost in doing so, in terms of both time and money. This comparison will provide us with a minimum valuation of the benefits of our technology to the system's curation costs. As these are all necessary steps, any productivity gains in this section will be realised generally.

In order to measure the likely future value and productivity that the system will exhibit, we need to estimate the likely cost of changing the schema and repurposing the data for new uses. For schema changes, we can simply use those changes that were introduced during the compilation of the published papers, measure how much it cost to achieve the required data quality levels after those changes and repeat the experiment with the new system. For data agility, we can measure the cost of making the data available for review with our tools and estimate the amount of effort that this would have required had our tools not been available – as this was the one part of the process of publishing the initial two Seshat papers that our tools were responsible for.

Together these three measures provide a very comprehensive way of evaluating and comparing the performance of the two alternative systems of interest – Seshat before and after ALIGNED tools were introduced. By looking at the likely evolution of scale of the system, we can provide reasonable estimates not only about the current value provided by our tools and methods, but the future value that they will provide.

Thus, in this document, we evaluate the system that was introduced by ALIGNED against the existing wiki-based system in terms of the cost of three particular tasks:

Curation

Extract, clean, and analyse a new section of wiki data to test a hypothesis so that the quality of the data and analysis is considered ready for publication.

Data Agility

Publish the data in a form that allows reviewers to evaluate all the data that went into the analysis, complete with sources, citations, uncertainty, and disagreement.

The major task that required data agility in Seshat was the publication of data for reviewers. The requirements were that the reviewers should be able to browse the dataset, view every individual datapoint with citations and indicators of uncertainty and disagreement, and that they, or any member of the public, could provide feedback on each individual datapoint. This was deemed important as a way of signalling to the research community that expert feedback and corrections were invited and taken seriously. This task was never competed with pre-ALIGNED methods, ALIGNED tools were entirely responsible for the dataset that was published to support the first major Seshat publication in April 2017. However, it is possible to provide a reasonably accurate estimate of how much it would cost using a standard software engineering approach:

- Develop and maintain custom program to transform Seshat extracted CSV into interactive website.

Model Agility

Change the schema by adding, removing, and changing the definition of variables, adding complex relationships between entities and returning the system to the quality level it had before the schema was changed.

For all schema changes, we need to carry out the following tasks:

1. Update Seshat code book wiki page
2. Copy updates to every wiki page that uses that section of the code book.
3. Update, test and deploy publication program to reflect changes in schema
4. Update and test script to transform CSV into analysis-ready format

Experimental Deployment

On 21 December, 2017, the first major paper was published utilising the full power of the Seshat data in the Proceedings of the National Academy of Sciences.¹⁴ ALIGNED tools were used to model, harvest, correct, improve, enrich, and publish the full data describing social complexity for 416 different historical polities that were used for the analysis. The analysis which formed the basis of the paper was carried out using the old – before ALIGNED – process. Upon publication, the published data were fully migrated to the Dacura platform and the processes were compared to evaluate the relative productivity, agility and quality – and the overall cost – of the process with and without the ALIGNED tools.

On 17 January, 2018, all of Seshat’s social complexity data were imported into a semantic model from the Seshat wiki. Figure 6.43 presents a summary of the results. The results are grouped into two groups – the WS30 group which represented the data that had been used to make the analysis and the Macrostates group which had not yet been analysed and therefore represented a more ‘raw’ version of the data with less effort expended in data quality control and analysis. The two groups were used to evaluate the impact of our tools when starting from different stages of an existing workflow.

¹⁴Turchin, P., T.E. Currie, H. Whitehouse, P. François, K. Feeney, D. Mullins, D. Hoyer, C. Collins, S. Grohmann, P.E. Savage, G. Mendel-Gleason, E. Turner, A. Dupeyron, E. Cioni, J. Reddish, J. Levine, G. Jordan, E. Brandl, A. Williams, R. Cesaretti, M. Krueger, A. Cecceralli, J. Figliulo-Rosswurm, P. Peregrine, A. Marciniak, J. Preiser-Kapeller, N. Kradin, A. Korotayev, A. Palmisano, D. Baker, J. Bidmead, P. Bol, D. Christian, C. Cook, A. Covey, G. Feinman, Á. D. Júlíusson, A. Kristinnsson, J. Miksic, R. Mostern, C. Petrie, P. Rudiak-Gould, B. ter Haar, V. Wallace, V. Mair, L. Xie, J. Baines, E. Bridges, J.G. Manning, B. Lockhart, P.-J. Tuan, A. Bogaard, and C. Spencer. 2017. “Quantitative historical analyses uncover a single dimension of complexity that structures global variation in human social organization.” Proceedings of the National Academy of Sciences of the United States of America. doi: 10.1073/pnas.1708800115. <http://www.pnas.org/content/early/2017/12/20/1708800115.full>.

#	Process Before	Service Provision Costs
1	Use validation tool to identify and fix syntax errors on each wiki page	Develop and maintain custom validation tool code Operating validation tool and fixing errors identified
2	Use scraper tool to extract data from wiki pages into CSV	Develop and maintain custom scraper tool service
3	Use script to transform CSV into required format for analysis	Develop and maintain transformation scripts
4	Analyse data and identify any remaining errors in the data.	Operate statistical analysis tool and inspect the results to identify anomalies, outliers, missing values and errors which might impinge upon the accuracy of the result.
5	If possible, correct all errors in CSV, complete the analysis, and copy corrections back to wiki from CSV	Identify and record all errors in the data and carry out known corrections. Manually copy all the corrections back to the wiki.
6	If impossible, deploy RAs to correct the data in the wiki manually and return to step 1.	Identify and record all errors in the data. Collect the corrections in the wiki. Carry out another iteration of the process.

Figure 6.43 Seshat: Comparison.

	WS30	Macrostates	Total
Polities	416	54	470
Variables	301,288	33,100	334,388
Non-empty variables	162,200 (54%)	10,779 (33%)	172,979
Imported variables	27,693 (17% of nonempty)	2,487 (23%)	30,180
Triples	995,580	75,716	1,001,296
Syntax Errors Detected	4	2	6
Semantic Errors Detected	218	117	335
Semantic Errors Corrected	86 (39%)	18 (15%)	104
Semantic Errors Remaining	132	99	231
Entity References	658	28	686
Correctly Imported	214 (33%)	5	219
Incorrectly Imported	444	23	467

These results then fed into our curation workflow, where provenance information was used to identify errors that had been detected but not corrected for manual correction. By far the biggest problem detection was in correctly identifying entity interlinks due to the lack of a consistent naming convention. Significantly, it was possible to identify 218 new errors that had evaded the human analysts and it was possible to automatically correct 39%. Nine days of RA labour was expended on fixing the problems identified, completed on January 29th 2018. The publication of the full dataset is

scheduled for the first week of February 2018, and the full evaluation results will be published in a paper that is under preparation.

6.3 Managing Data for the NHS

6.3.1 Introduction

Oxford University Software Engineering researchers have been involved in four separate projects involving health research data, which have made extensive use of the Model Catalogue and components of Semantic Booster.

In the first application, the National Institute for Health Research (NIHR) commissioned the Health Data Finder – an online tool for discovering national healthcare datasets (Figure 6.33). These datasets primarily contain routine hospital data for audit and economic reasons, but may be made available to researchers in academia and industry with appropriate governance approval. The datasets are maintained by a number of separate organisations, and so data users wishing to discover data and request access may have to make a number of requests, often with inconsistent results (Figure 6.32).

In the second application, the NIHR Health Informatics Collaborative, five of the largest teaching and research hospital trusts in the country have been asked to share routine clinical data in five therapeutic areas. Each trust maintains data to differing standards and semantics, and rather than unifying data to a lowest common denominator, sites are asked to build their own data warehouses for a federated data store. Users of the data can make a request to the hospitals, and data can be linked and unified on a per-usage basis, taking the research purpose into account. This allows hospitals to maintain ownership of their data and ensures data quality is as high as possible for any given research study. The model catalogue is used to document national data standards in each of the therapeutic areas, alongside local differences for each hospital trust. Models in the catalogue are used as the source for the generation of MS-Word documentation, and for data transfer specifications in the form of XML Schema.

A third application of the catalogue, and related technologies, has been made in the UK 100,000 Genomes Project. As in the Health Informatics Collaborative project, the catalogue has been used for the collaborative, iterative development of models for sample tracking, cancer and rare disease data models, and the generation of non-technical documentation, XML Schema, and also Case Report Forms, compatible with a commonly used clinical trials management system. In the pilot phase of the project, the models in

the catalogue were also used to generate relational databases, sufficient for storing data collected according to the specification.

The fourth application of the ALIGNED tools is in the construction of a data warehouse for Oxford University Hospitals Foundation Trust. This instance of the catalogue acts as a detailed asset register for the hospital, detailing field-level metadata about databases and spreadsheets of patient data around the hospital, as well as describing dataflows and message-passing between systems, and specifications for audit and research datasets. It is planned that these models can be used as part of a data-science platform for the trust: allowing clinical researchers to request data, and be automatically guided through governance processes, as well as provided with the data presented in a secure environment.

6.3.2 Use Case

6.3.2.1 Quality

In all four applications, reuse of existing data without detailed documentation can be problematic: researchers are unable to make good use of the data without understanding its semantics: linkage between datasets may be inaccurate; transformation of data into different formats may be incorrect; interpretation of statistical results is error prone. In the Health Data Finder example, such data reuse is minimal: researchers do not know what data may be available to them; different providers may return inconsistent results on data governance, and data must be re-interpreted each time, which may result in costly errors.

In similar projects preceding the Health Informatics Collaborative and 100,000 Genomes projects, collecting comparable data from multiple hospitals has proven difficult. Precise specifications have been hard to produce, mechanisms for data capture and transfer have been manually programmed, often by non-technical domain experts, and inconsistencies have resulted in data that are often incomplete, incomparable, or completely unusable.

6.3.2.2 Agility

The quality and accuracy of data documentation is difficult to maintain during an iterative process. In all the health data research projects, datasets are continually evolving, data specifications are continually being improved. Without careful version management and automation, it is very easy for the documentation to get left behind.

Similarly, software artefacts must keep pace with the changes in requirements: changes to the data or the software specifications must invoke updates to the XML schema, database schema, or Case Report Forms. Manual coding slows the iteration process, which in turn can result in outdated or inaccurate specifications.

Productivity

Domain experts find it difficult to provide documentation or simple modelling because of the technicalities involved: XML schema and Case Report Forms require specialist technical knowledge: domain expertise is often left out, or modelling is undertaken poorly.

Implementing efficient database structures requires a lot of repetitive works: implementation of a domain class will involve a familiar pattern of tables, association tables, keys, and indexes. Such work is time-consuming and error prone, yet ripe for automation.

Data scientists looking to reuse health data currently spend a lot of time searching for usable datasets, often requiring long periods of interaction where inventories and documentation are not available online. Applying for governance, asking technical questions, and retrieving data in a suitable format often require further time and energy. Interpretation and curation of the data is a typically manual task, which may be repeated and reproduced by every scientist receiving a data extract.

6.3.3 Architecture

In each project, ALIGNED technologies are being used in slightly different ways.

In the NIHR Health Data Finder, the model catalogue is the central resource, holding the master copy of models and documentation. A REST-based API provides services used by the front-end website that provides shopping-cart and dataset overview functionality. Metadata is imported into the catalogue by means of a bespoke spreadsheet-based format, which is suitable for domain experts and data curators to populate.

In the NIHR Health Informatics Collaborative, each site hosts its own instance of the model catalogue, documenting their own data landscape: a data warehouse, source patient record systems, research systems and local data flows. A central installation of the catalogue contains the shared data specifications, along with local variations, and relevant national specification.

Local catalogue installations can automatically import the latest version of the central models, and the central catalogue is used to generate XML schemas for use by all partners.

In the UK, 100,000 Genomes Project, the architecture of the pilot is of particular interest: information is provided by the hospitals in the form of XML, matching a schema generated by the Model Catalogue, or manually through online Case Report Forms, hosted in a system called OpenClinica. Information is extracted via an ETL process from OpenClinica, and combined with a shredded form of XML, and stored in a matching relational database, generated by a component of Semantic Booster.

Finally, the architecture of the OUH data warehouse follows a similar pattern to the right-hand-side of Figure 6.44. Almost 100 local databases and data specifications are modelled within the catalogue, along with the design for the main data warehouse. The catalogue is used to document field-level metadata, summary metadata, and dataflows, and this information will be used in the construction of research data extracts and for generating hospital auditing and service improvement metrics.

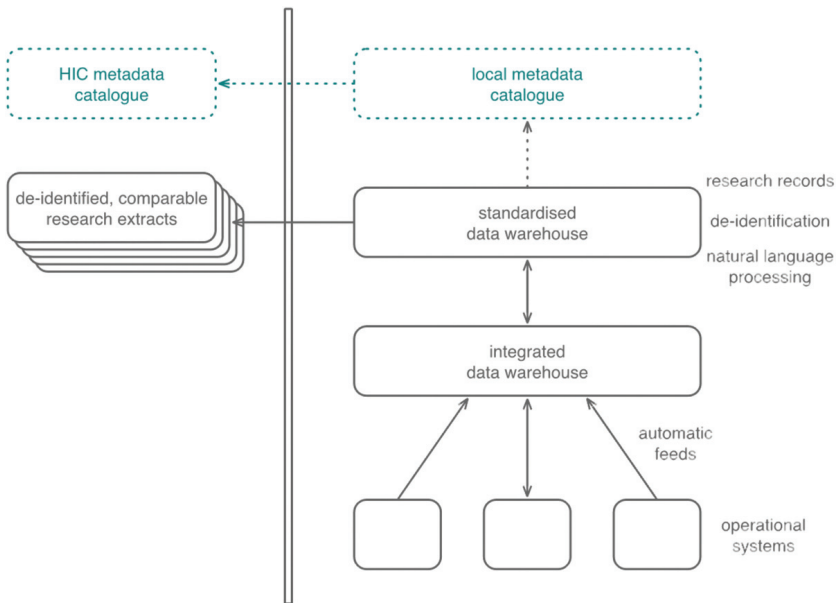


Figure 6.44 Health Informatics Collaborative system architecture.

6.3.4 Implementation

6.3.4.1 Model catalogue

Whether you have a small spreadsheet, or a large federated data warehouse, the key to making the most of your data is understanding its semantics. In order to share your data with others, to reuse it for a different purpose, or to link it to other data stored elsewhere, you have to know what it means. At its simplest, this is just knowing a datatype, and having a description of how those data have been collected. But you may also know how the data have been curated, where it was created in the context of business processes, or how it relates to recognised standards. To do this at scale requires automation: tools that can do the hard work for you and allow the rest to be done collaboratively. Our metadata catalogue tool provides a common framework in which to store descriptions of data alongside data standards, terminologies and dictionaries, providing common reference points by which to describe data.

The catalogue is able to automatically import models – structured descriptions – from relational databases, XML schema, spreadsheets, and UML diagrams. A collaborative editing environment allows the iterative development of models in a clean, simple fashion: just suitable for domain experts to really focus on the things that are important. The catalogue facilitates reuse of data models: parts of one model can be dropped into another. This will make it easy to reuse data in the future and can help to proliferate data standards. Describing data is made easy: links to existing descriptions are automatically suggested; classes of data can be described in a single place, and creating new versions of models maintains any semantics already expressed. Finally, the models can then be exported in a variety of different formats: as relational databases for storing data, or as XML schema for data transport, or as forms for collecting data from scratch. Models for software engineering tools can also be generated – for example by generating specifications for our Semantic Booster tool, we enable the complete, automatic generation of working information systems. In this way, the catalogue can be used as an IDE for an agile, model-driven approach to software- and data engineering. Figure 6.45 shows the catalogue interface.

6.3.4.2 NIHR health informatics collaborative

We now illustrate the advantage of the catalogue with three case studies. In the first, the Oxford team have led the coordination of the Health Informatics Collaborative – a project funded by the National Institute for Health Research

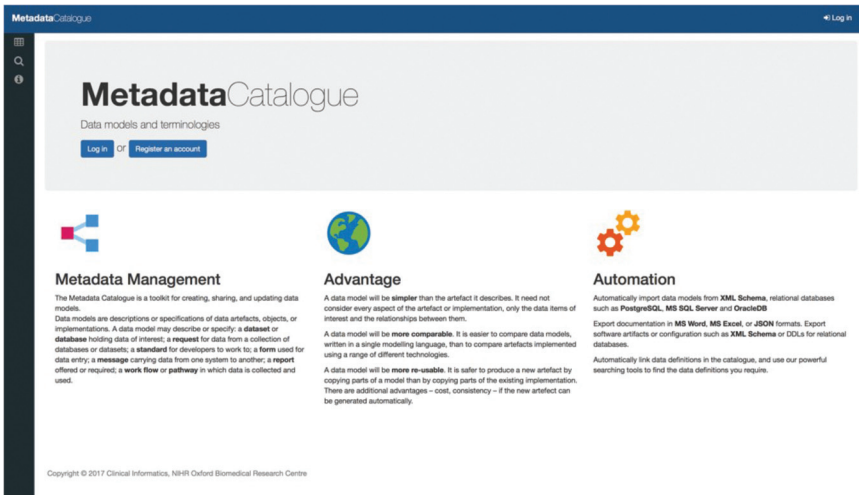


Figure 6.45 The front page of the catalogue interface.

to promote the sharing of healthcare data in the UK. Five of the largest research hospitals in the country – across London, Oxford, and Cambridge, were asked to share routine clinical data on five therapeutic areas: in critical care, ovarian cancer, acute coronary syndromes, hepatitis and renal transplantation. Clinicians at the hospitals were asked to collaborate on the definition of a new dataset, suitable for addressing a wide range of research issues within each clinical specialty, and the hospitals were asked to share anonymised data matching these data specifications. The metadata catalogue provided tools for collaborative editing of dataset specifications, maintaining older versions for reference. XML schema were generated for data transfer between the sites, and the catalogue was able to generate Excel spreadsheets for documentation.

One of the main problems with data sharing amongst healthcare providers in the UK is that each site may record their data points differently. Here the catalogue provided another useful feature – allowing each site to document their own variations, and details of any transformations required to translate data from one format into another. Each of the five data models had, on average 250 data points of interest, and we were able to map relationships between the NHS's own data dictionary, as well as existing standards and audits in each area. The project has created combined datasets for the first time in these areas of clinical interest, enabling new research and, in some cases, better treatment. Figure 6.46 shows the project in action.

The screenshot shows a web application window titled "NHC Aggregate View". At the top, there are tabs for "ACB", "CAN", "HEP", "ICU", and "TRA". Below these is a header "Ovarian Cancer". The main area contains a table with columns for "NHC Identifier", "Name", and several data columns grouped under "OUH", "GSTT", "IMP", "OUH", and "UCL". Each of these groups has sub-columns labeled "A", "B", "C", "D", and "E". The table contains 13 rows of data, each representing a different clinical data point. The cells contain numerical values (1, 2, 3, 4) and dashes, indicating data presence or absence across different sites. Some cells are highlighted in green or pink.

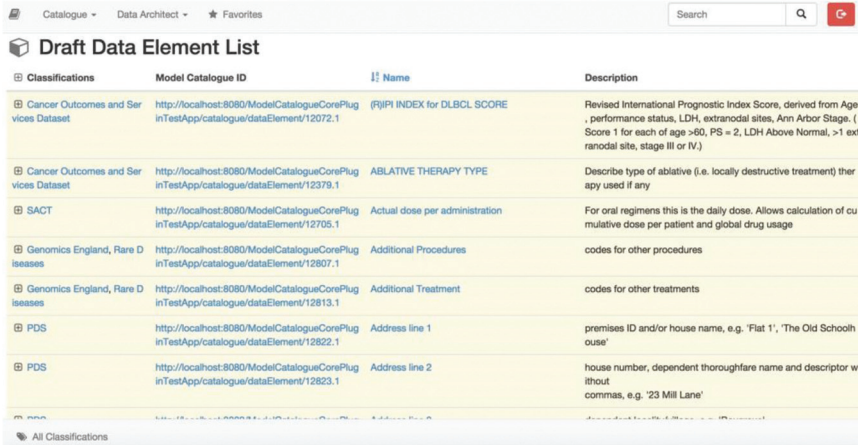
NHC Identifier	Name	OUH					GSTT					IMP					OUH					UCL				
		A	B	C	D	E	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
NHC_CAN_1	NHS NUMBER*	2	2	3	1	4*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_2	LOCAL PATIENT IDENTIFIER*	2	2	3	1	4*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_3	NHS NUMBER STATUS INDICATOR CODE	2	2	1	1	1	4	4	4	4	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
NHC_CAN_4	PERSON BIRTH DATE	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_5	ORGANISATION CODE (CODE OF PROVIDER)	2	1	1	1	1	4	4	4	4	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_6	PRIMARY DIAGNOSIS (ICD)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_7	DATE OF DIAGNOSIS (CLINICALLY AGREED)*	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_8	DATE OF RECURRENCE (CLINICALLY AGREED)*	2	2	1	3	2*	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_9	PERSON FAMILY NAME	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_10	PERSON GIVEN NAME	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_11	PATIENT USUAL ADDRESS (AT DIAGNOSIS)	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_12	POSTCODE OF USUAL ADDRESS (AT DIAGNOSIS)	2	2	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
NHC_CAN_13	PERSON GENDER CODE (CURRENT)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

Figure 6.46 Data comparison in the Health Informatics Collaborative.

UK 100,000 Genomes Project

In the last couple of years, Oxford has also been involved in a large genetics programme – the UK 100,000 Genome Project. The project was set up to revolutionise personalised medicine in the UK, starting with the whole genome sequencing of NHS patients with key forms of cancer, as well as patients and family groups with rare inherited diseases. Again, the catalogue was used by the scientists to develop new datasets for routine clinical data, and brand new, bespoke models for each of nearly 200 rare diseases. The metadata catalogue was again used to generate XML schemas for data transfer, but also for electronic case report forms, compatible with a widely used clinical trials management software. These forms were built to include terms from existing medical ontologies, including the Human Phenotype Ontology, and SNOMED CT. For the pilot studies, the catalogue was also used to build databases, used to store the clinical and sample-tracking data on submission. These databases were entirely generated by the data model: a change to the model in the catalogue resulted in a new schema for the database, along with an appropriate data upgrade. Figure 6.47 shows the catalogue.

The project is now halfway to completion and would not have succeeded without the catalogue's provision of a central data model. The national Genomics Medicine Centres rely on the catalogue as the specification for prospective data collection, and those interpreting the data rely on its descriptions to make sense of the data collected. Initial results include confirmed diagnoses for patients with unspecified rare-diseases, and the refinement of lab processes for processing DNA samples at scale.



Classifications	Model Catalogue ID	Name	Description
Cancer Outcomes and Services Dataset	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12072.1	R/RP INDEX for DLBCL SCORE	Revised International Prognostic Index Score, derived from Age, performance status, LDH, extranodal sites, Ann Arbor Stage. (Score 1 for each of age >60, PS = 2, LDH Above Normal, >1 ext ranodal site, stage III or IV.)
Cancer Outcomes and Services Dataset	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12073.1	ABLATIVE THERAPY TYPE	Describe type of ablative (i.e. locally destructive treatment) therapy used if any
SACT	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12705.1	Actual dose per administration	For oral regimens this is the daily dose. Allows calculation of cumulative dose per patient and global drug usage
Genomics England, Rare Diseases	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12807.1	Additional Procedures	codes for other procedures
Genomics England, Rare Diseases	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12813.1	Additional Treatment	codes for other treatments
PDS	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12822.1	Address line 1	premises ID and/or house name, e.g. 'Flat 1', 'The Old Schoolhouse'
PDS	http://localhost:8080/ModelCatalogueCorePlugInTestApp/catalogue/dataElement/12823.1	Address line 2	house number, dependent thoroughfare name and descriptor without commas, e.g. '23 Mill Lane'

Figure 6.47 Data elements in the UK 100,000 Genomes Project catalogue.

NIHR Health Data Finder

A final example of where the catalogue has been providing benefit is the UK's Health Data Finder. This instance of the metadata catalogue, commissioned by the National Institute for Health Research, provides a portal for healthcare researchers in industry and academia, allowing them to discover national datasets. These datasets, collected at scale across the whole health service, are primarily collected by a number of different bodies for commissioning or audit purposes, but are of great value because of their size. There was no easy way to inform potential users exactly what those datasets contained, and the process for requesting data was time-consuming and prone to error. The catalogue now provides element-by-element descriptions for over 3,000 data points, across more than 20 datasets. It stores summary metadata and usage information, sufficient for researchers to understand whether the data will help them answer a particular question before starting to request any of the valuable data. We are currently streamlining the process for requesting data, by using the catalogue as a 'shopping cart', allowing researchers to select a set of data points to request, and generating queries to return those data points once sufficient governance checks have been made. The shopping cart is shown in Figure 6.48.

The catalogue has provided a number of benefits to healthcare projects across the UK, but is continuing to be developed and extended. Figure 6.49 shows the catalogue in the Health Data Finder. We are increasing the range of models that can be imported into the catalogue, and we are continuously

The screenshot shows the 'Model Catalogue' interface with a 'Draft' shopping cart containing several data models. The models listed are:

- Hospital Episodes Statistics (HES) - Admitted Patient Care
- Hospital Episodes Statistics (HES) - Outpatient
- Hospital Episodes Statistics (HES) - Adult Critical care
- Hospital Episodes Statistics (HES) - A&E attendance
- Patient Reported Outcome Measures - PROMS
- Mental Health & Learning Disability Data Set - MH/LDDIS
- Diagnostic Imaging Dataset - DID
- National Cancer Data Repository - NCDR
- Primary Care Data Set - CPRD GOLD
- ONS - Mortality dataset
- Secondary Uses Services - SUS
- Primary Care Data Set - GPES

The 'Selection Items' table below shows details for each model:

Name	Type	Description	
4 character concatenated diagnosis <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	Provides a concatenated string of all diagnosis codes at a... more	✗
Age on admission <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	A patient's age, in years, at the date of admission.	✗
Administrative & legal status of patient <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	Many NHS hospitals have private wards where private... more	✗
Discharge episode flag <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	Codes in this field indicate whether the episode is a... more	✗
Age at end of episode <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	This derived field contains the patient's age in whole... more	✗
Legal status classification code at start of episode <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	Required for all patients with a hospital provider spell... more	✗
Postcode Found <i>Hospital Episodes Statistics (HES) - Admitted Patient Care / Patient</i>	dataElement	Field confirms if postcode is valid	✗

Figure 6.48 An example shopping cart in the Health Data Finder.

The screenshot shows the 'Metadata Catalogue' interface for the 'Mental Health & Learning Disability Data Set'. The details are as follows:

Metadata Catalogue | Models | Search | About | [Return to Health Data Finder](#)

Models > Classifications

- HES - Admitted Patient Care
- HES - Outpatient
- HES - Adult Critical care
- HES - A&E attendance
- Patient Reported Outcome Measures
- Mental Health & Learning Disability Data Set**
- Referral (REFD)
- Mental Health Team Episode (TEAMEP)
- NHS Day Care Episode (DMVEP)
- Consultant-led episode (CONEP)
- Acute Home-based Care Episode (BECAREEP)
- Mental Health NHS Care Home Stay Episode (BHCAREHOMEP)
- Hospital Provider Spell (PROVSPELL)
- Inpatient Episode (INPATEP)
- Ward Stay within Hospital Provider Spell (WARDSTAYSP)
- Delayed Discharge (DDP) (DDOCHDNG)
- Clinical team (CLNTTEAM)
- Staff Details (STAFF)
- Care Coordinator Assignment Start Date
- Responsible Clinician Assignment Start Date
- Health Care Professional Contact (HPCONT)
- Review (REV)
- Mental Patient Index (MPI)
- Psychosis Services (PSYCHOSIS)
- Employment Status (EMP)
- Accommodation Details (ACCOMMD)
- Secondary Diagnosis (SECDSG)
- Primary Diagnosis (PRIMDSG)
- NHS Day Care Facility Attendance (DCAVFT)
- Proposed Characteristic Disability (DISABILITY)
- HANDS for People with Learning Disabilities (HANDSLD) (HANDSLD)
- Sleeping Status Table (SMDONSDSTATUS)
- Conditional Discharge (CONDCHD)
- Commissioner History (COMBHR)
- Period of Seclusion (SECLURN)
- Assaults on Patient (ASSALUT)
- Use of Restraint (RESTRAINT)
- Self-Harm (SELHMR)
- Home Leave (HOMELVLE)
- Absence without Leave (AWOL)
- Level of Admission (LDA)
- Electro Convulsive Therapy (ECT)

Mental Health & Learning Disability Data Set New

dataElement
Last Update: 2016-12-22 09:29:09

Author: HSCIC

Organization: HSCIC

Description: The Mental Health & Learning Disability Data Set (MH/LDDIS) contains record-level data about the care of adults and older people using secondary mental health, learning disabilities or autism spectrum disorder services at: * NHS hospitals * community clinics * NHS-funded activity in the independent sector

Type: DataSet

Classifications

Metadata Annotations Links Change History

Metadata Table

Key	Value
Dataset Group	Mental Health & Learning Disability Data Set
Dataset	
Organisation	HSCIC
Type of data	Secondary Care - National Mandated data sets
Description	The Mental Health & Learning Disability Data Set (MH/LDDIS) contains record-level data about the care of adults and older people using secondary mental health, learning disabilities or autism spectrum disorder services at: * NHS hospitals * community clinics * NHS-funded activity in the independent sector
Coverage	NHS-funded treatment from all specialist practitioners
Geography	England
Volumes	Approximately 1m spells per month
Availability	* from 2003 * from 2006/07 - for linkage
Events	Spell of care by NHS-funded providers of adult secondary mental health and learning disability services

Figure 6.49 The model catalogue in the Health Data Finder.

improving the usability for non-technical domain experts – including graphical editing tools, automated search and suggestion, and new visualisations. We are especially interested in using these models as the basis for MDE, and so plugins are being written to generate or configure software components so that reuse of models can really instigate reuse of data. Figure 6.50 shows an example of catalogue metadata. The catalogue has been extensively used in the domain of healthcare, but is fundamentally nonspecific to any particular domain – our work with the ALIGNED partners is helping us prove the

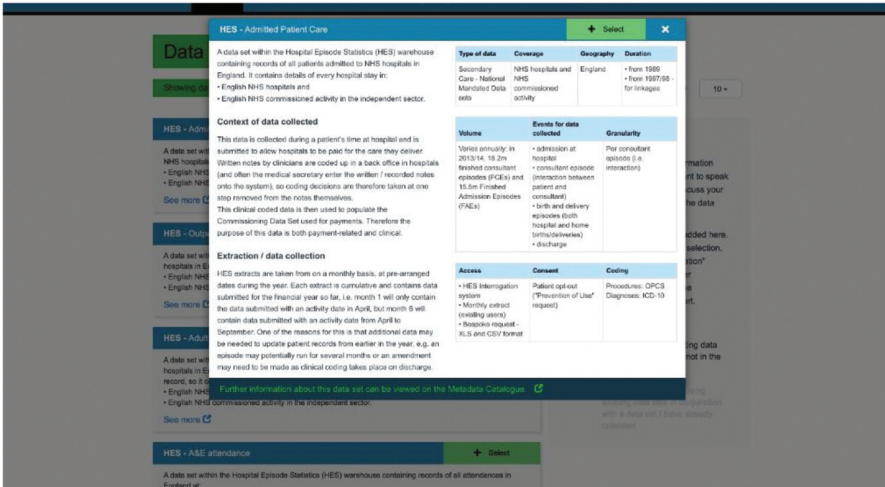


Figure 6.50 Dataset metadata in the NIHR Health Data Finder.

technology in other domains. Our experiences with the tool show that it can be invaluable for software engineers and data engineers alike.

6.3.5 Evaluation

In the Health Data use case, the Model Catalogue has been deployed in four main projects: The UK 100,000 Genomes project, the NIHR Health Data Finder, the NIHR Health Informatics Collaborative (HIC), and the Oxford Biomedical Research Centre’s data warehousing activity. In all four projects, the catalogue has provided functionality that was not previously available, or automated tasks that were previously undertaken by hand. The utility of the ALIGNED tools can be measured by their usage: if the tools are used frequently, then they provide a valuable service.

Across the four projects, the model catalogue was primarily used for two separate use cases: firstly the management and documentation of existing data assets – allowing potential data users to search and discover datasets of interest; secondly the collaborative development and publication of new data standards – reusing existing definitions where available. The Health Data Finder and Oxford BRC Data Warehouse projects are primarily focussed on the cataloguing of existing datasets or databases, and the Health Informatics Collaborative and UK 100,000 Genomes Project are primarily concerned with the development and publication of new data standards in a number of medical therapeutic areas.

For each of the measures: “Productivity”, “Quality” and “Agility”, we will consider each of the four projects and assess the impact made in these areas. In most cases, quantitative measures of improvement are not easily obtained: the Health Data use case was a late addition to the project and the relevant baseline measures were not taken; however, the use of the catalogue in all four cases does not replace any existing functionality or tool provision – originally any software or data engineering tasks were carried out by hand or not at all.

6.3.5.1 Productivity

The NIHR Health Data Finder was set-up to be a single portal for researchers – both academic and in industry – to find out about existing national audit datasets that can be requested for research purposes. Before the introduction of the Model Catalogue, this could be a painful process: the datasets are held by one of a number of public health bodies: NHS Digital, Public Health England, the Clinical Practice Research Datalink, the National Institute for Health Research, and the Medical Research Council. Each maintained their own documentation for the datasets, usually stored in non-computable formats, and, in general, not made publicly available. If a research data user required data, they would have to first find out whether such data existed, and telephone a help desk to ask any questions about the data; detailed questions could take weeks to be answered. Requesting the data would require a different governance process for each provider, and data would be provided in different formats by each provider. All data are anonymised before being conferred: if data from multiple providers were required to be linked before anonymisation, this would increase the complexity of this largely manual process.

The Model Catalogue provides a solution to some of these problems, and forms part of a greater plan to streamline all data requests. The catalogue provides a single portal where all datasets are described, datapoint-by-datapoint, with information about the scope, coverage and completeness for each dataset. Information pertaining to ‘frequently asked questions’ is stored alongside each data element, and adherence to national standards is recorded. As well as advanced functions for browsing and searching, the catalogue provides a ‘shopping cart’ function which allows users to compile requests made up from multiple datasets.

The time saved by the use of the catalogue tool is hard to quantify, as each request is different. However, the site has been used more than 4,200 times in 2 years since its launch, with an average of six visitors per day. Of these,

approximately 40% are returning visitors, indicating some degree of success on their first visit. The average ‘session’ duration for all visitors is well over 3 min, suggesting that a lot of users are taking the time to browse and explore. Although the number of visitors has dropped since the first launch of the site, the numbers remain stable.

In the Oxford BRC Data Warehousing project, a team of developers are building a large warehouse of patient data, extracts of which will be made available to local researchers for specific purposes. In order to maintain an asset register and to provide documentation to potential users, every data source and data flow is being documented. Before the introduction of the catalogue, this documentation would have been maintained in a series of spreadsheets and shared (perhaps in a source control system) to allow collaboration. The catalogue provides plugins that automate the transcription of database metadata, and descriptions can be collaboratively edited via the online interface – a vast improvement to productivity. There are currently 12 developers and data engineers using the catalogue – some on a daily basis – and allowing access to Oxford University researchers is planned in 2018.

In the Health Informatics Collaborative, and in the UK 100,000 Genomes project, a key output is the development of new data standards – to facilitate the transfer of clinical data from a number of different hospitals to a centralised location. In such projects, collaboration is required from a range of different people: those with clinical expertise to assess the availability of data; those involved in research to assess the requirements for each data point, and technical people at each hospital who can assess the feasibility of providing data. Previously such collaboration may not have happened or taken place via email and teleconferences; with the use of the Model Catalogue such collaboration is much easier, and can reduce the number of iterations required to reach a viable data specification.

In the UK 100,000 Genomes project, complex models for Cancer and over 200 Rare Diseases have been developed and published, iterating through a number of intermediate versions. In the NIHR HIC project, models for five therapeutic areas have been developed: originally using spreadsheets and email; latterly using the catalogue. The catalogue has reduced the amount of communication required and simplified the task of development and documentation of the model; a further five new therapeutic areas are to be addressed with new models in the NIHR HIC project, during the first quarter of 2018.

The screenshot displays the 'Angiography' model details in the NIHR HIC Model Catalogue. The interface includes a search bar at the top left, a left-hand navigation menu with categories like 'Classifications', 'Angiography', and 'Table', and a main content area with fields for Description, Parent Hierarchy, Multiplicity, and Classifications. Below these fields is a 'Data Elements' table.

Name	Data Type	Description
DATE OF CATHETERISATION OF HEART	Text	Indicates the DATE of the CATHETERISATION OF HEART
MAXIMAL STENOSIS	Text	Indicate the maximal stenosis of epicardial vessel - This... more
NUMBER OF VESSEL DISEASE (NVD)	Text	Number of coronary vessels with stenosis >50% - If NVD... more
SEVERE CAD	SEVERE CAD	If >50% lesion in any vessel between two admissions - This... more
TIME OF CATHETERISATION OF HEART	Text	Indicate the TIME of the CATHETERISATION OF HEART

Figure 6.51 Screenshot from the NIHR HIC Model Catalogue.

6.3.5.2 Quality

In the UK 100,000 Genomes and NIHR HIC projects, as well-documented data standards, key outputs are software components to allow the storage and transfer of data according to the standard. Without automation, it would be very easy for mistakes to be made in the development of tools such as XML schema or database schema: differences between the standard and the tools could result in data not being transmitted or stored correctly. The plugins developed for the Model Catalogue allow these components to be generated automatically. In the early stages of the HIC project, when such a manual process was in place, discrepancies arose frequently, and this caused delays and frustration as errors had to be corrected, new standards or tools re-tested, published and distributed. The introduction of the catalogue has seen a complete reduction in these errors, and also reduces development effort (Figure 6.51). In the UK 100,000 Genomes project, further components were required to configure off-the-shelf software, and suitable plugins were developed to ensure that these also remained consistent with the standards.

In the Oxford BRC Data Warehousing project, data quality can be improved by allowing those entering the data to see the descriptions of the intended values – so they know how to complete fields correctly – or to see the data already submitted – in order to fix any problems with existing data. In its current state, running metrics on the existing data has identified a number of potential issues with the data and other local reporting, and so the Model

Catalogue has become a useful tool for the reporting and discussion of these issues.

In the Health Data Finder project, an improvement has been made, not in the quality of the actual data, but in the linking and usage of the data. With detailed descriptions of every data point, researchers are better able to make decisions on how to use the data – in many cases preventing mistakes in analysis, or, where previously the semantics of data points were unknown, preventing researchers having to collect new data from scratch to ensure its validity for the particular purpose.

6.3.5.3 Agility

One of the key advantages of the catalogue product is the ability to create new versions of a model with ease, ensuring that all participants can be kept up-to-date, and by using plugins to generate software components, updates to a data model can be reflected in changes to the related software much more quickly. In the NIHR HIC project, this is an essential requirement: the XML schema required for transferring data between sites can be made available as soon as the new data model is finalised – giving technical staff the maximum amount of time to adapt to the new model. Previously, delays in the generation of XSDs (and subsequent fixing of any errors), could delay the timely collection of data. A similar improvement has been made in the UK 100,000 Genomes project, where without the use of the Model Catalogue, manual approaches to collaborative model evolution, publication and software development would result in a much slower turn-around time.

In the Health Data Finder and Oxford BRC Data Warehousing projects, the key notion of agility is in the time taken to update the documentation in response to a new version of the database schema. Again, the plugins have proven invaluable in this respect: the importer plugins can automatically import the new structures, and existing descriptions can be copied, meaning that minimal effort is required from domain experts.

6.4 Integrating Semantic Datasets into Enterprise Information Systems with PoolParty

6.4.1 Introduction

PoolParty Semantic Suite is the SWC's platform for enterprise information integration based on Linked Data principles. Since it was created, the product has evolved to include entity extraction from unstructured information. To

align product development with ongoing technology trends, market monitoring and trend scouting features have been incorporated. Atlassian Confluence is used to support the requirement engineering process while Atlassian JIRA is used for issue tracking, including an external system for customers.

The developers of the SWC's software have numerous sources of information that is relevant to their product development role – bugs, feature requests, usage information, and so on. They would like to ensure that the information relevant to any particular development task is made available to the relevant developers in as timely, well-structured and meaningful way as possible, regardless of the source. Customers of PoolParty would like to integrate a variety of models, schemata, ontologies and vocabularies into their PoolParty knowledge bases. In many cases, they do not have a deep understanding of semantic technologies and would benefit from as much assistance as possible in understanding what they need to do to integrate their models into PoolParty.

To support and document the development process, SWC operates installations of Atlassian Confluence and JIRA. Confluence is used for drafting, specifying and discussing new features and requirements in a text-based format which is only structured visually with headings and paragraphs. Most requirements captured in Confluence follow a defined structure: they declare the high-level goal (or summary), which is a description of the functionality the application should provide so that the requirement is met. The requirements document breaks down this description into multiple “user stories” which are detailed descriptions of how the application should behave from a user perspective. They also add preconditions, acceptance criteria and test scenarios so that the responsible developer can identify what changes need to be performed and infer JIRA tickets for each of them. A requirements document also defines various stakeholders, i.e., people and their responsibilities and roles they fulfil in the course of processing the requirement.

JIRA defines a data schema to hold the details of each ticket, like type, description, priority, or assignee. On the most general level, tickets (also sometimes called issues in this section) are assigned to various “spaces”. A space is used to classify issues by project (e.g., LOD2 or ALIGNED), product (PoolParty Thesaurus Manager PPT or PoolParty Extractor PPX) or general kind (ideas, which are “nice-to-have” features or improvements for which it is not yet decided if and how they will be implemented). Each ticket can only be assigned to one space and the space, to some degree, also influences the properties that can be assigned to a ticket. For instance, valid types that can be assigned to a ticket are, e.g., “bug”, “task”, “epic” or “story”

in the PPT space while “epic” or “story” cannot be assigned to tickets in the PoolParty Support space. Besides the affected software components, status, resolutions methods and much more, also metadata is attached to the ticket like creation and last-updated date. The properties mentioned above which are relevant for querying in the ALIGNED use case(s) are modelled in the Design Intent Ontology (DIO) by OxSE, which is used for publishing the data held by Confluence and JIRA as RDF.

6.4.2 Problem Statement

The developers of the SWC’s software have numerous sources of information that is relevant to their product development role – bugs, feature requests, usage information, and so on. They would like to ensure that the information relevant to any particular development task is made available to the relevant developers in as timely, well-structured and meaningful way as possible, regardless of the source. Customers of PoolParty would like to integrate a variety of models, schemata, ontologies and vocabularies into their PoolParty knowledge bases. In many cases, they do not have a deep understanding of semantic technologies and would benefit from as much assistance as possible in understanding what they need to do to integrate their models into PoolParty.

6.4.2.1 Actors

<i>Role</i>	<i>Description</i>
<i>PPT Developer</i>	performs software development work on the PoolParty platform
<i>PPT User</i>	uses PoolParty
<i>PPT Taxonomy Developer</i>	responsible for developing taxonomies for PoolParty
<i>PPT Admin</i>	responsible for administering PoolParty services
<i>Requirements Engineer</i>	responsible for defining and maintaining software requirements
<i>SWC System Administrator</i>	responsible for administering SWC assets

The requirements on which the PoolParty use case was based are detailed in Appendix A.

6.4.3 Architecture

Figure 6.52 shows the different roles (orange figures), tools (green rectangles), repositories (cylinders), and files (parallelograms) involved in the PoolParty architecture and workflows. On the left side, the diagram describes

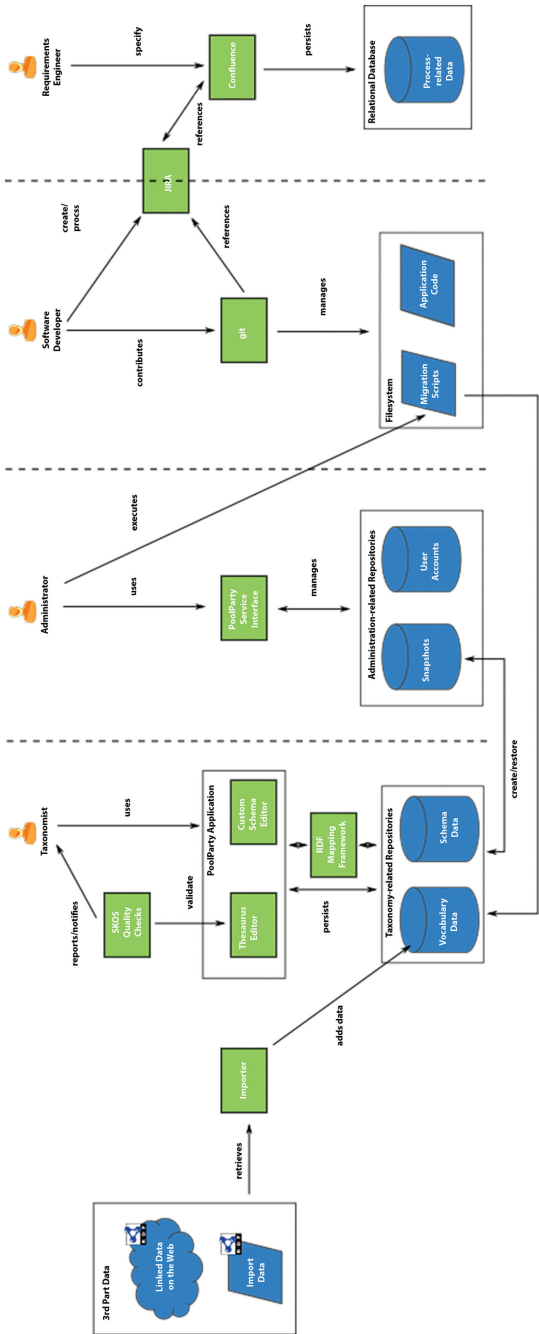


Figure 6.52 PoolParty Architecture.

the direct interaction of the customers (taxonomists) with the PoolParty application. The taxonomist creates controlled vocabularies using both the thesaurus editor and the custom schema editor. Currently, there are two components where data consistency needs to be satisfied: (i) when persisting vocabulary and schema data to the underlying triple store using a custom SWC-developed RDF Mapping Framework and (ii) when changes to the controlled vocabulary are performed which violate certain quality criteria (validated by the SKOS Quality Checks component). The RDF mapping framework converts instances of annotated Java domain classes into an RDF representation and vice versa. However, data consistency violations between the triple store(s) and the application via the RDF mapping framework can occur because in the application code, the framework is sometimes bypassed and data changes are written directly to the store. Furthermore, changes to the domain classes may require migration scripts which can easily be forgotten to develop and run. Also note that the data importer component which retrieves data either from the LOD cloud or from imported files currently persists these data directly to the triple store, which in many cases violates data consistency requirements.

6.4.4 Implementation

The demonstrator system consists of four components, which we shortly outline in the following paragraphs:

- Consistency violation detector
- RDFUnit test generator
- PoolParty integration
- Notification adaptations

6.4.4.1 Consistency violation detector

We implemented the consistency violation detector as a separate component that can be either invoked on the command line or integrated into PoolParty as a library. It takes as input the id(s) of the consistency violation check(s) it should detect as well as an arbitrary number of RDF files that contain all necessary data for performing the check(s). All these RDF data are then added to a local in-memory OpenRDF repository, together with the RDF definition of the SKOS data schema. All but one of the identified consistency violations can be detected by using SPARQL queries over the provided

RDF input files. The one constraint where SPARQL queries do not suffice is the validation of external links. This is done by a Java algorithm that dereferences all URIs (that do not reference localhost) and checks if the HTTP response code indicates an error (i.e., other than 200).

6.4.4.2 RDFUnit test generator

Test cases for RDFUnit are expressed in RDF as resources of, e.g., type `http://rdfunit.aksw.org/ns/core#ManualTestCase`. Our demonstrator system can generate the test cases for RDFUnit automatically, based on the SPARQL queries we defined for each data consistency check. Currently, four of the 16 data consistency violation checks can be automatically converted to RDFUnit tests. The RDFUnit test cases can then be executed on the in-memory repository mentioned above and a HTML report page is generated by RDFUnit which shows the results (success or failure) of each test case.

6.4.4.3 PoolParty integration

For the demonstrator, we integrated the consistency violation detector into PoolParty's data import functionality. The current implementation checks for violations of any of the 16 identified data consistency constraints. Therefore, it first collects the data of all linked projects, the project metadata, and custom schema data and passes it to the consistency violation detector. The generated textual report is then displayed to the user, along with the option to view the HTML page that has been generated by the RDFUnit test run.

6.4.4.4 Notification adaptations

We improved the rsine notification system¹ which has been originally developed in the course of the LOD2 project (see Section 5, Improved Notifications, or the project's GitHub page for additional information) to:

- Be transaction-aware: Due to improvements on how PoolParty invokes data changes, rsine can persist them as a transaction. This enables us to write easier and more powerful notification subscriptions.
- Support of project management, custom schema, and user repositories.

Until now, only changes to the taxonomy project repository were communicated to the notification service. We changed that so that it is also possible to subscribe for changes to the project management, custom schema and user management repositories to, e.g., receive notifications on creation of new projects, new custom classes, or new PoolParty user accounts.

6.4.4.5 RDFUnit

RDFUnit is integrated in PoolParty RDF Validation for performing constraint checks. The checks are defined as RDFUnit test cases using RDF. These test cases can also be run by RDFUnit independently of PoolParty on external data. For each of the constraint checks, there is an RDFUnit test case which is based on a SHACL constraint or a SPARQL query that identifies resources that cause violations.

UnifiedViews is an ETL tool for RDF data developed as part of the PoolParty semantic suite. Using this tool, we extracted data from Atlassian Confluence and JIRA and transformed it into RDF using a DPU developed for ALIGNED. The transformed data are annotated with the PoolParty Knowledge.

Graph using the extractor DPU and finally similarity scores are calculated based on the annotated data.

The Issue Integration feature is integrated in PoolParty product, which allows user to automatically create JIRA support tickets whenever an internal server error occurred in the application.

Similarity scores are calculated on development artefacts using the annotations of the PoolParty Knowledge Graph Thesaurus as a basis. Two algorithms are implemented that represent a lexical and a graph-based approach to similarity.

Graph Search, a faceted search application and part of the PoolParty product, is used to analyse the development artefacts. We integrated similarity retrieval into GraphSearch to find duplicate bugs and relations between issues.

6.4.4.6 Validation on import

General Description

Currently, users can import any RDF data into a PoolParty thesaurus project. In the best case, invalid data just lingers in the triple store where PoolParty stores all the data it operates on and consumes memory or hard disk space. However, these data also can cause problematic behaviour such as inconsistency in the user interface and a corrupt data model, manifesting in fatal exceptions in the PoolParty Thesaurus Editor. We can identify three different PoolParty functionalities where data consistency is required:

- Basic internal operations: The thesaurus editor expects certain properties for the various controlled vocabulary resources, such as concepts or concept schemes

- Schema-specific: SKOS or other data schemas impose custom restrictions on the data or encourage conformance with best practices that are not formally stated
- Reasoning: PoolParty asserts and expects class membership information to controlled vocabulary resources and interprets them with constraint semantics.

Addressed Challenge

The main challenge is to match the imported data, which follow the open world assumption with the local data model required by PoolParty. This is basically a challenge each application that consumes open data from the Web faces. Because these data are very volatile, efficient methods have to be in place that allow transition of data scraped from the Web into a meaningful local representation that can be further processed by the application's business logic.

Identify Sample Set of Data Consistency Violations

We can break down this challenge to a set of sub-goals we want to solve in the course of the ALIGNED project:

- Provide full coverage of data consistency constraints
- Identify repair strategies
- Invoke repair strategies and fix constraint violations either automatically or based on user input

Proposed Approach

We plan to support the import use case with a two-step semi-automatic scenario: in the first step, the imported data must be checked against PoolParty's internal data model and requirements on the data and any non-conformance must be reported. In a second step, users should have the option to adjust the imported data in order to fulfil PoolParty Thesaurus Editor's requirements. Based on the kind of data consistency violation, various repair strategies may be invoked. Some violations can be fixed automatically and some require additional input from the user. It should also be possible to fix similar kinds of consistency violations in one go so that it is possible to deal with a large number of violations.

Identified Data Consistency Constraints

For demonstrating the problem domain and working towards the implementation of an approach that tackles the addressed challenge, we first focussed on the sub-goals 1 and 3. We extracted 16 data consistency constraints, i.e., requirements for RDF datasets so that they match the internal PoolParty Thesaurus Editor (PPT) data model. Violations of these constraints can vary in severity: some constraints must never be violated (ERROR), some can be tolerated (WARNING) and some are just of informative value (INFO). For each of the identified consistency constraint, we propose one or more repair strategies that describe possible ways to fix the dataset.

We implemented a tool (usable both at the command line as well as a library for integration into existing applications) that checks provided RDF data against violation of these constraints. A current development branch of PoolParty makes use of this tool and displays a report if constraint violations on imported data were detected. Four of the consistency constraints listed above have also been formulated as RDFUnit test case and can thus be integrated into existing test suites.

ID	Constraint	Description	Severity	Resolution Strategies
br	Bi-directional Relations	If a resource A is related to a resource B by a property p and if p has an inverse property p', then the statement that B is related to A by p' must also be manifested in the data.	ERROR	Add complementary statement Remove relation
cd	Concept Deletion	In order for PPT to recognise deleted concepts, these concepts must be marked with owl:deprecated, must not have asserted any type information, and must contain information in the history graph for being properly displayed in the application.	ERROR	Remove other facts that are not asserted by owl:deprecated Remove "owl:deprecated true" fact
cta	Concept Type Assertion	Concepts must have the type skos:Concept asserted because no RDFS inferencing is performed in PPT.	ERROR	Add (infer) missing type declarations
cdl	dcterms Creator Literal	Using URIs for dcterms:creator to describe skos:Concepts and skos:ConceptSchemes in PPT leads to error message	ERROR	Convert provided creator agent to literal Replace with some default literal

Continued

ID	Constraint	Description	Severity	Resolution Strategies
dta	Direct Type Assertion	Concepts having asserted a class using <code>swcs:appliedType</code> must also be instances of this class.	ERROR	Add missing type statement Remove resource
elv	External Link Validity	Outgoing links from a thesaurus to another dataset on the Web may not be resolvable anymore.	INFO	Prompt user for replacing URI with “valid” link Apply resolution strategy suggested when dereferencing the URI Remove affected statement
hc	Hierarchical Consistency	Each resource of type <code>skos:Concept</code> must have a resource linked by <code>skos:broader</code> or <code>skos:topConceptOf</code> in the vocabulary namespace. Each resource of type <code>skos:Concept</code> must have at least one path (via <code>skos:broader/skos:topConceptOf</code>) to a resource of type <code>skos:ConceptScheme</code> in the vocabulary namespace.	ERROR	Prompt for parent resource Add to some existing default parent resource Remove (do not create or ignore) concept
lam	Label Ambiguities	Identical concept labels may indicate duplicate concepts.	WARNING	Remove Label from one concept (prompt user for which one) Merge Concepts Add descriptive note (prompt user for text input)
lav	Label Availability	Resources of type <code>skos:Concept</code> must have assigned exactly one Literal in the default language, using the predicate <code>skos:prefLabel</code> . Resources of type <code>skos:ConceptScheme</code> must have assigned exactly one Literal in the default language, using the predicate <code>rdfs:label</code> .	ERROR	Auto-generate label (based on URI, timestamp, increment, from parent/related...)

(Continued)

Continued

ID	Constraint	Description	Severity	Resolution Strategies
lpc	Linked Project Consistency	If two PPT projects are linked to each other, each of the referenced resources must exist.	WARNING	Remove Link Restore Data (i.e., create new local concept with deleted concept's label)
sc	Schema Compatibility	Detect statements using resources from namespaces that are not included in the default PoolParty schemas or in schemes that are available as custom schemas. Such statements would not be visible within PoolParty and may lead to unwanted side effects.	WARNING	Enable relevant schemas in PPT Ignore statements
sdr	Schema Domain Range Match	Domain and range axioms on a property are interpreted as constraints – that is, a property with specified domains (using swcs:domain) A and B can only be used in triples with resources that are instances of A or B. Likewise for swcs:range.	ERROR	Apply missing type(s): either one (prompt user which one) or all Remove relation
tpc	Type Propagation Collections	All concepts that are members of a collection which is instance of a class (using the property swcs:appliedType) also are instances of this class.	ERROR	Assert missing Types Remove from collection
tph	Type Propagation Hierarchical	Concepts that are part of a hierarchy (using skos:broader properties) and one of the parents (e.g., a resource being an instance of skos:ConceptScheme and skos:Concept) have either a type asserted (using swcs:appliedType for skos:ConceptSchemes) or propagated (using swcs:propagateType) must also be instances of this class.	ERROR	Assert missing types Remove from hierarchy

Continued

ID	Constraint	Description	Severity	Resolution Strategies
upl	Unique Preferred Labels	A concept must have at most one preferred label per language tag (SKOS integrity constraint)	ERROR	Remove one preferred label (prompt user which one) Add disambiguation information as notes (prompt user to supply them) Add disambiguation information as parenthesis to label (bad practice) Extract new concept (prompt for broader or insert as sibling) Remove concept
ut	Unsatisfied Type	Concepts must either be instances of skos:Concepts or instances of classes that are assigned directly or by type propagation.	ERROR	Remove type assertions Remove affected resources Import type as custom class

Detailed Process Description

In the following, we show how import data validation is implemented in a proof-of-concept branch of PoolParty:

(1) Accessing the RDF Data import functionality of PoolParty: the newly adapted import dialog provides an option for checking the imported data for conformance against the consistency constraints (Figure 6.53).

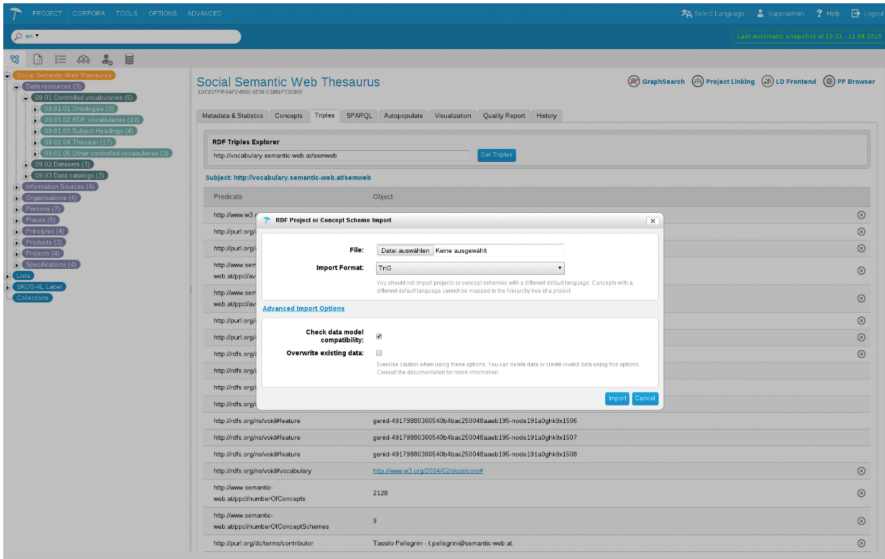


Figure 6.53 Import dialogue.

(2) Report on the resources that violate certain consistency constraints (Figure 6.54).

```

TestExecution: http://rdfunit.aksw.org/data/results#67de6a18-2aef-11b2-8029-525400e63664
  Dataset file://somefile
  Test suite http://rdfunit.aksw.org/data/testsuite#67de6a1a-2aef-11b2-8029-525400e63664
  Test execution started 2015-09-11T09:01:00.451Z
  -ended 2015-09-11T09:01:01.828Z
  Total test cases 4
  Succeeded 3
  Failed 1
  Timeout / Error T:0 / E:0
  Violation instances 1

Results
Level Message Resource Test Case
WARN Missing assertion of owl:inverseOf property http://foac.host:8080/gnedsweb/import_3 http://poolparty.biz/rdgndpdt_test_bidirectionalrelationshierarchical
    
```

Figure 6.54 Consistency constraint violations as reported by RDFUnit.

6.4.5 Results

By providing a demo implementation of an import validator, we found that RDF datasets can be checked against the identified data consistency constraints, either by using SPARQL or by a hybrid approach, processing a subgraph generated by SPARQL with custom Java algorithms. Based on the query results, reports containing the resources that violate consistency constraints are created. We also found that the consistency constraints

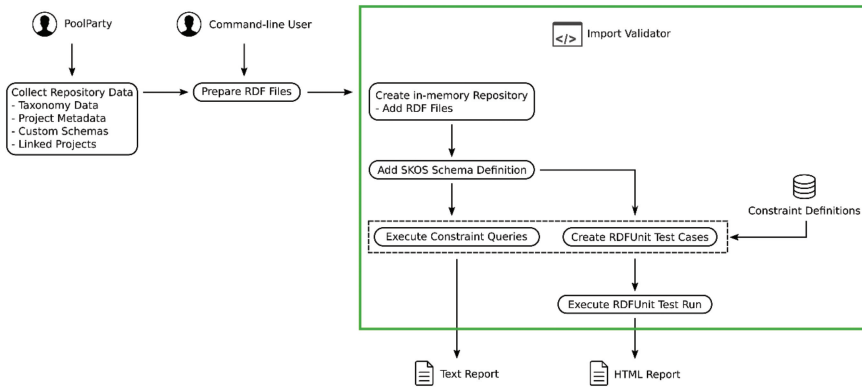


Figure 6.55 High level technical overview.

which can be solely expressed using SPARQL (i.e., no custom Java algorithms for validation are needed), can also be expressed as test cases for RDFUnit.

For identification of the above described consistency constraints, we analysed the algorithms PoolParty uses internally for creating, processing and persisting a controlled vocabulary. While this is efficient for getting an initial set of constraints, we cannot retrieve a complete set of consistency constraints which covers all error cases this way. The reason is that a formal model of the data that PoolParty operates on does not yet exist. Therefore, the consistency constraint checks must be manually crafted in SPARQL, independent from the algorithms creating or accessing the data. As a consequence, the checks constitute an additional entity that must be maintained in sync with changes to the application logic.

Figure 6.55 illustrates the workflow for checking RDF data for conformance to the PoolParty data model. The PoolParty integration collects data from various sources necessary to evaluate potential consistency violations. Alternatively, command line users of the Import Validator can prepare RDF files and pass them to the Import Validator Component (green frame), which applies the constraint definitions to this data and outputs a textual or HTML report that contains violation information.

6.4.5.1 RDF constraints check

Figure 6.56 shows the constraints checks integrated using RDFUnit. When importing data into a PoolParty project, the constraint checks are performed,

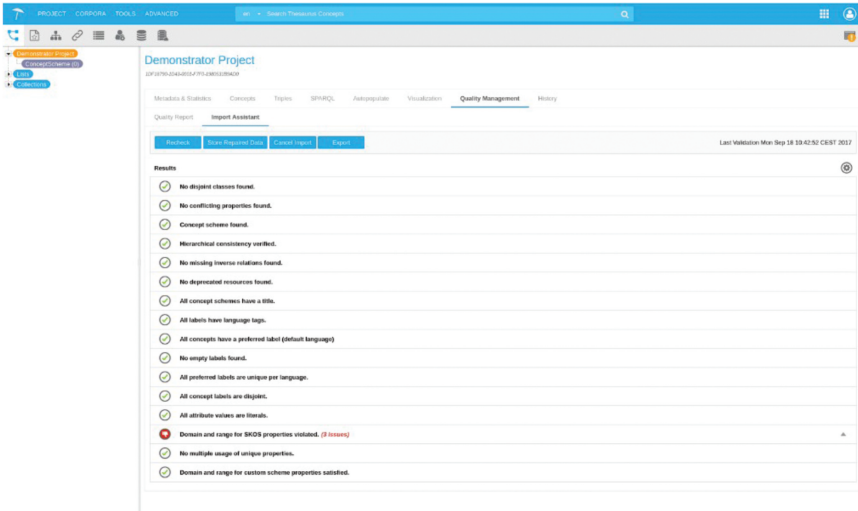


Figure 6.56 RDF validation conformance checks.

and a result list is presented to the user who outlines all the violations that have been detected. For the first release of PoolParty containing the RDF Validation, we defined a minimal set of 13 constraints so that imported data are required to conform with PoolParty to operate normally. In later releases, this set will be extended by quality checks to assist with data modelling. The declarative approach taken for defining the constraints ensures easy maintainability and extendability of the RDF Validation for future releases.

The user can browse through the detected constraint violations, select a repair strategy for each of them and apply the repair to the data (Figure 6.57). When all violations are repaired and conformance is achieved, the user can transfer the imported data into the project without the risk of application failures caused by inconsistent data.

6.4.5.2 RDF validation

The first part is the RDF Validation, which is integrated into the PoolParty product to support consistency within the application's data storage.

Data within PoolParty projects have to follow conformance rules for the application to work correctly. Usually, data are modified by the application itself, and the conformance is therefore given naturally. However, it is possible for users to import arbitrary RDF into the project. These data have to

The screenshot displays the 'Results' section of the RDF Validation tool. At the top, there are navigation tabs: 'Results', 'Data Properties Data', 'Constraint Issues', and 'Export'. The 'Results' tab is active, showing a table of violations. The table has three rows, each representing a different constraint violation. Each row includes a source URI, a constraint name (e.g., 'skos:narrower', 'skos:broader'), and a target URI. To the right of each row, there is a 'Repair' button. The first row shows a 'skos:narrower' constraint between 'http://localhost:8080/SolarSystem/Default/Planets' and 'http://www.w3.org/2002/XML/Schema/string'. The second row shows a 'skos:broader' constraint between 'http://localhost:8080/SolarSystem/Haumea' and 'http://localhost:8080/SolarSystem/Default/Planets'. The third row shows a 'skos:narrower' constraint between 'http://localhost:8080/SolarSystem/Default/Planets' and 'http://localhost:8080/SolarSystem/Haumea'. The interface also shows a 'Last Validation Mon Sep 18 10:42:52 CEST 2017' timestamp and a 'Results' header with a red circle icon and the text 'Domain and range for SKOS properties violated. (2 Issues)'.

Figure 6.57 Repair strategy for the constraint check.

be checked and eventually corrected to conform to the PoolParty application. To ensure this conformance, the import component of PoolParty was extended with an RDF Validation component. It is responsible for checking the imported data based on a set of defined constraints and reporting the results. The user is then given the opportunity to correct the import by using one or more presented repair strategies that will manipulate the data so they satisfy the constraints. Bulk repair options are also given for constraints where it is appropriate to do so. Furthermore, general quality checks can be done on the data that do not interfere with PoolParty's operations, but represent data modelling problems and would therefore be of interest to the Taxonomist.

For performing the constraint checks, the RDF Validation has integrated RDFUnit. The checks are defined as RDFUnit test cases using RDF. These test cases can also be run by using RDFUnit only and therefore can be used independently of PoolParty on arbitrary data. Also, the maintainability of the constraint checks is high because of the declarative approach of the test case definition using RDF. Changing the checks does not require changes to the application's code. The repair strategies and other metadata are also defined as RDF and extend the RDFUnit test cases for an integrated representation of validation and repair. For each of the constraint checks, there is an RDFUnit test case, which is based on a SPARQL query that identifies the resource that causes the violation. Each check also defines repair strategies that can be applied to fix the violation. The information needed for the repair strategies to determine changes that have to be done can be retrieved using a constraint specific query that returns the context of the violation as RDF statements. The combination of constraint, context and repair strategies is represented as an extension of the RDFUnit test case. The component implementing the test cases is designed to be independent of PoolParty and can be used separately. PoolParty integrates it to present the RDF Validation as an application feature.

When importing data into a PoolParty project, the constraint checks are performed, and a result list is presented to the user that outlines all the violations that have been detected. The user can browse through these violations, select a repair strategy for each of them, and apply the repair on the data. When all violations are repaired and conformance is achieved, the user can transfer the imported data into the project without the risk of application failures caused by inconsistent data.

For the first release of PoolParty containing the RDF Validation, we defined a minimal set of 13 constraints that are mandatory to conform with PoolParty to operate normally. In later releases, this set will be extended by quality checks to assist with data modelling. The declarative approach taken for defining the constraints ensures easy maintainability and extensibility of the RDF Validation for future releases.

RDF Validation: The user imports an RDF file into a project. A list of constraint violations is shown and explained. Constraint violation details are opened and the constraint details are shown. The repair strategy is executed. Another constraint violation is shown and repaired. Afterwards, all the violations have been resolved. It is explained that a save import is now possible (Figure 6.58).

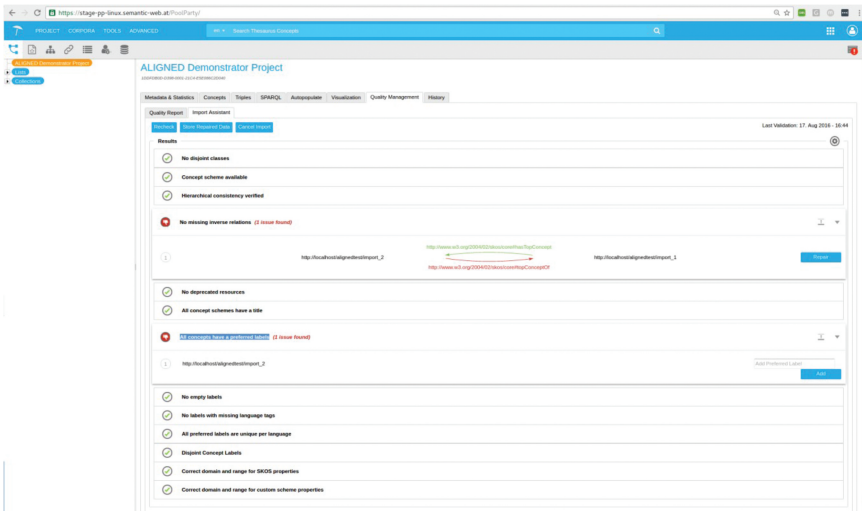


Figure 6.58 RDF Validation Screenshot.

6.4.5.3 Improved notifications

General Description

During the LOD2 project, SWC developed rsine,¹⁵ a publish/subscribe system that allows users to register for data changes in an RDF triple store. In the demonstration system, we reuse and adapt rsine to work with a current version of PoolParty and extend it to support additional notification types as required by Wolters Kluwer.

Addressed Challenge

The LOD2 technology stack¹⁶ consists of multiple tools that cover the whole Linked Data life cycle. It encompasses, among others, storage, quality analysis and exploration utilities that target problem domains that also affect PoolParty. LOD2 project partners needed a way to better integrate their solutions, and being notified on data changes between stack components was one of the project goals. Therefore, the notification systems were required to be:

- easily integratable into existing stack components, and
- flexible enough to support notifications which can be adjusted to meet the component's purpose and data model.

Approach

Rsine runs as a stand-alone server and can be controlled by a REST-like interface. It can be configured against a Managed RDF Store (accessible by a SPARQL endpoint), which holds all data a LOD2 stack component works on. Addition and deletion of triples to this managed store are detected by the Change Handler. It forwards these changes to the Changeset Service, which enriches them with additional metadata such as timestamps using a standard ontology¹⁷ and persists them into an in-memory Changeset Store.

We currently support two different types of change handlers:

- Integration with the managed store: an external component, e.g., a Virtuoso VAD extension¹⁸ or transaction log parser¹⁹ detects triple changes in the underlying Virtuoso triple store.

¹⁵<https://github.com/rsine/rsine>

¹⁶<http://stack.lod2.eu/blog/>

¹⁷<http://vocab.org/changeset/schema.html>

¹⁸<https://github.com/rsine/rsineVad>

¹⁹https://github.com/GeoKnow/trx_parser

- Integration with the stack component: The stack component (e.g., Pool-Party) is responsible for announcing all data changes to rsine using API calls.

To subscribe for notifications, users can submit Subscription Documents to the rsine server using the API. These are RDF documents, containing information about

- The change patterns (as SPARQL query) the user should be notified about,
- A notification message,
- Additional information the notification message should contain (fetched from the managed store using SPARQL), and
- Contact information (e.g., email address, log file) where the notification should go to.

A complete description of the information a change document should contain can be viewed at the project’s GitHub page. Figure 6.59 shows the architecture of the notification system.

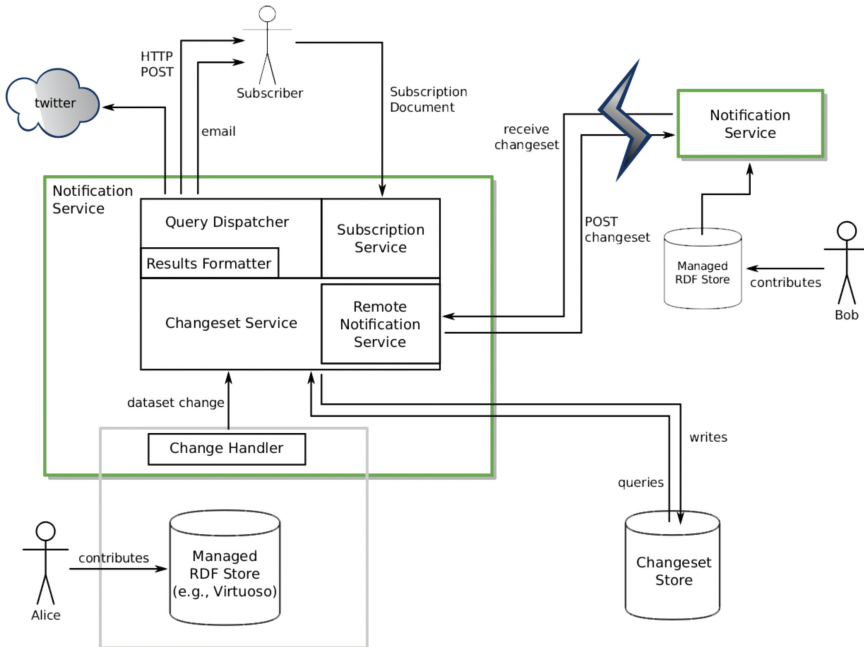


Figure 6.59 Improved notification system.

Improvements

For the demonstration system, we revised and improved rsine to meet the new notification requirements of WKD. We adjusted a current version of PoolParty, with the following change handler improvements:

- Dataset changes are now transaction aware: if a PoolParty action (e.g., creation of a document) creates or removes multiple triples at once, these are combined and stored as a single RDF changeset representation. This feature required us to adjust the rsine API and break compatibility with older rsine versions
- Support for other repositories than the current vocabulary repository. The change handler can now communicate data changes introduced to the project metadata repository, the custom schema repository and the user accounts repository

Notifications

These changes allow us to express and implement new types of notifications that were not possible with other rsine PoolParty integrations before. On project creation, for example, a notification containing the user who created the project and the project's name can be disseminated. Notification can also be done when creating, changing or deleting classes, attributes or properties of custom schemas in PoolParty or creating and deleting PoolParty user accounts. In the following, we provide an abbreviated example (we omitted the prefix declarations) subscription document that logs a message if a new user has been created.

```
<http://example.org/aligned/new\_account> a rsine:Subscription;
  rsine:query [
    dcterm:description "Notification user account creation";

    spin:text "SELECT ?userName (GROUP\_CONCAT(?auth; separator=', ' )
AS ?auths) WHERE \{
  ?cs a cs:ChangeSet;
    cs:createdDate ?csdate;
    cs:addition ?userAdd;
    cs:addition ?userAuth;
    cs:addition ?userInfo.

  ?userAdd rdf:subject ?user;
    rdf:predicate rdf:type;
    rdf:object swcu:User.

  ?userAuth rdf:subject ?user;
    rdf:predicate swcu:grantedAuthority;
    rdf:object ?auth.
```

```

?userInfo rdf:subject ?user;
           rdf:predicate swcu:username;
           rdf:object ?userName.

FILTER (?csdate
>'QUERY_LAST_ISSUED'^^<http://www.w3.org/2001/XMLSchema#dateTime>
)
GROUP BY ?userName HAVING (STRLEN(?auths) > 0)
";

rsine:formatter [
  a rsine:vtlFormatter;
  rsine:message "A new user named
'$\bindingSet.getValue('userName').getLabel()' with the roles
'$\bindingSet.getValue('auths').getLabel()' has been created.";
]
];

rsine:notifier [
  a rsine:loggingNotifier;
].

```

Detailed Process Description

The command `java -jar ./rsine-cmd.jar` starts the rsine notification server, accepting notification subscription documents on port 2221

The notification subscription document can be registered at the server using the command: `curl -X POST -d @"create_user_account_subscription.ttl" --header "Content-Type: text/turtle" http://localhost:2221/register`

Rsine detects the event and adds a notification to the log: 13:17:26.258 [qtp524197922-12] INFO e.l.r.d.n.logging.LoggingNotifier – A new user named ‘aligneduser’ with the roles ‘PoolPartyUser, PoolPartyAdmin, Public’ has been created.

Note that notifications can also be configured to be sent to an email address by adding this snippet to the notification subscription document:

```

rsine:notifier [
  a rsine:emailNotifier;
  foaf:mbox <mailto:c.mader@myhost.at>
];

```

Results

We found that the new notification subscription documents, covering project metadata, custom schema changes or user account management, were easy

to implement on the rsine side. However, we had to put more effort into adapting the Change Handler components, which are part of the newly created PoolParty-ALIGNED branch, to support the new notification types. The reason for this is that PoolParty internally organises vocabulary data (i.e., the SKOS representation and some metadata of a taxonomy project), project metadata, custom schemas and user account information in different RDF repositories, and only the vocabulary data can be accessed by a SPARQL endpoint. This has two major consequences:

- We had to integrate the Change Handler code into PoolParty's data persistence logic for each repository,
- We currently cannot cover all information that should be contained in the notification messages. For example, if adding a new class to a custom schema, the notification message can only contain the name of the new class, not the schema name it has been added to or the user who created it.

However, future releases of PoolParty will only use a single repository for the data described above and organise it into different named graphs. This will allow us to efficiently query the data and also to formulate more powerful queries, aggregating knowledge of each named graph. Therefore, PoolParty's rsine integrate will also profit from disseminating more detailed and useful notification messages to the subscribers.

6.4.5.4 Unified governance

The Unified Governance tool is used to harvest data from the tools used in the PoolParty development life cycle. The data are transformed into RDF and integrated using ALIGNED vocabularies to create unified views for supporting the development process.

The Unified Governance tool will support three use cases for the trials:

- Search over the integrated RDF software development data
- Computing similarity for software development artefacts based on a combined graph-based and text-based approach
- Statistical analysis of the software development process

The sources of development data used for the tool are Atlassian Confluence and Atlassian JIRA.

Atlassian Confluence is used for requirements engineering, organising ideas from team members, providing documentation of research projects, and publishing of technical information. Atlassian JIRA is used for issue

management as part of the SCRUM-based software development process. It is also used as a ticketing system for customers to report issues. Both of these tools are used for integrated software development process, but they are not integrated with each other. This has to be done by humans as part of the process to synchronise the information. It includes manual linking of requirements in Confluence to JIRA issues and linking duplicate JIRA issues together. Generally, an integrated and interlinked view of requirements and development artefacts is needed. With the Unified Governance tool, this can be achieved automatically.

The tool retrieves the information from both Confluence and JIRA and transforms it into RDF based on the ALIGNED metamodel vocabularies DIO and DIO-PP. This has to be done on a regular basis to have up-to-date information to work on. Therefore, we use Unified Views, an Extract Transform Load (ETL) tool supporting RDF data processing, for periodic retrieval and transformation of the development data. Having integrated the data as RDF, we can query it using SPARQL. The queries can make use of the underlying metamodels to improve the results. Furthermore, SPARQL-based applications can be put on top of the triple store to support querying, filtering and faceted search.

During the integration process, the generated RDF data are annotated with concepts using a PoolParty Thesaurus. These concepts can support search applications. They can also be used as a basis for computing similarities between artefacts based on the hierarchical graph structure of the Thesaurus. A graph-based approach can leverage the underlying knowledge model and provide semantic similarity for the development artefacts. We decided to use a combined method of text-based and graph-based similarity to benefit from both approaches and improve the results. The results of the similarity computation can be applied to several tasks. First, we can automatically identify developments issues that correspond to requirements and semi-automatically link them. Second, we can identify similar requirements and ideas that should be organised together, but appear distributed in the system. Third, we can identify duplicate issues in JIRA, which is important to prevent the duplicate reporting of bugs. We can identify duplicates before an issue is submitted, inform the user about it and eventually prevent the creation of the issue.

The RDF data of development artefacts can be used for a statistical analysis of the development process. The results can then be used to apply improvements. They can be used as a reference basis for future time estimations of efforts. Flaws in the development process can be identified by analysing performance decreases. Development efforts and reported bugs can

be analysed for deviations from the expected values. Statistical data will be visually presented in form of diagrams as part of the search application.

Unified Governance: A UnifiedViews pipeline is used to extract data from Atlassian Confluence and JIRA (Figure 6.60). The data are transformed into RDF and integrated using ALIGNED vocabularies to create a unified view on the development data for supporting the development process. The transformed data are then annotated with concepts from the PoolParty Knowledge Graph. Similarity between development artefacts is calculated using a lexical and a graph-based approach in combination.

Unified Governance Search: The faceted and autocomplete search application on top of the Unified Governance data is explained in detail (Figure 6.61).

Unified Governance Similarity: The similarity computation as part of the search application and the use cases are explained.

Unified Governance Statistics: The statistical analysis and the visualisation are explained.

Issue Integration: Data inconsistencies in PoolParty can be caused by application error or can be caused by user by importing the data in PoolParty without doing constraint checks. These types of inconsistencies which cannot be handled by PoolParty can be reported by using the Issue integration feature (Figure 6.62). It allows users to configure a JIRA instance and report the issue automatically to PoolParty support (Figure 6.63). The log file is also automatically attached to the issue.

Graph Search: The faceted search which is used for managing development artefacts. It also provides a recommender UI where users can see the similarity between different issues and requirements. By using this recommender, users can find duplicate bugs, similar stories, and the requirements, which are associated with specific bugs.

Users can search for issues and see the details about it (Figure 6.64). GraphSearch provides a selection of similarity algorithms that were integrated for this use case to calculate similarities between development artefacts (Figure 6.65).

6.4.6 Evaluation

6.4.6.1 Measuring overall value

PoolParty is a software product provided to customers on premise or as a cloud service. Although the value can be measured by commercial success, the improvements done to both the application's features and the development process cannot be easily quantified. Data curation for PoolParty during

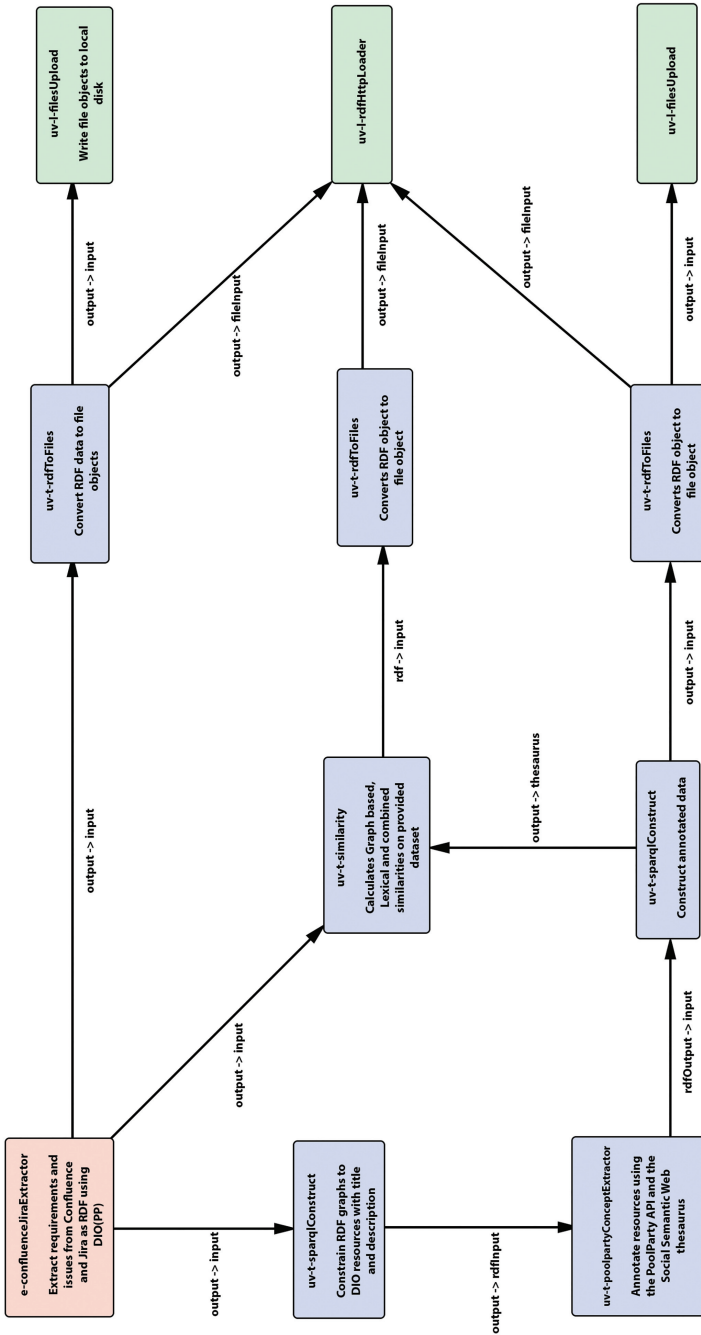


Figure 6.60 UnifiedViews pipeline for PoolParty use case.

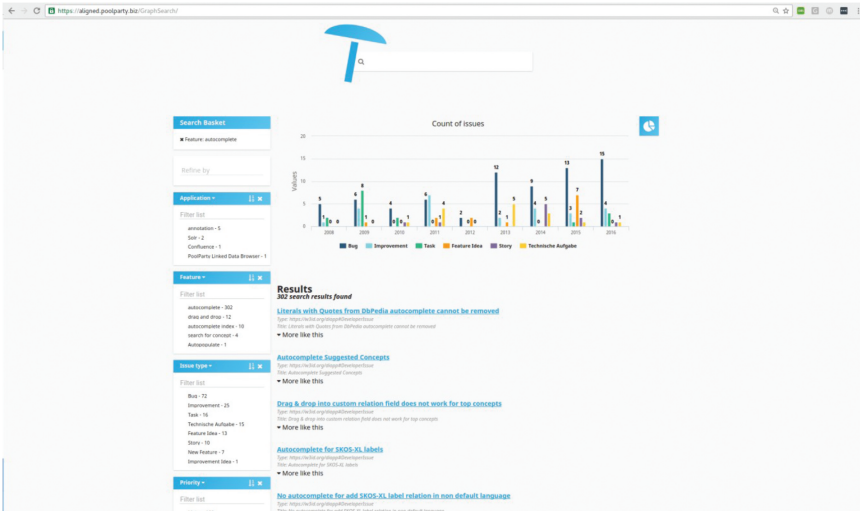


Figure 6.61 Unified Governance Search.

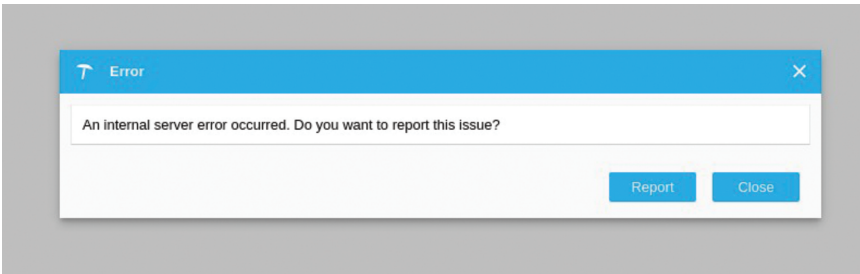


Figure 6.62 Issue Integration reporting dialogue.

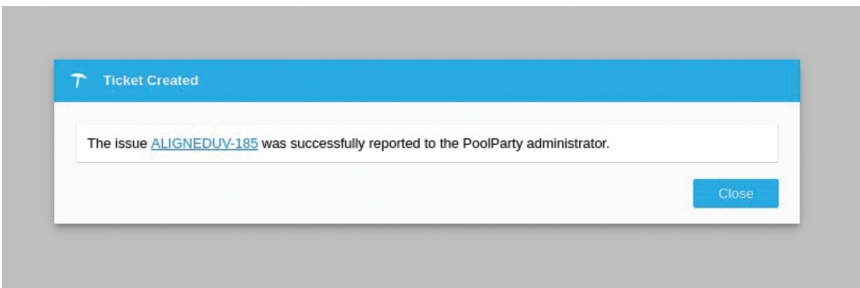


Figure 6.63 Issue Integration created dialogue.

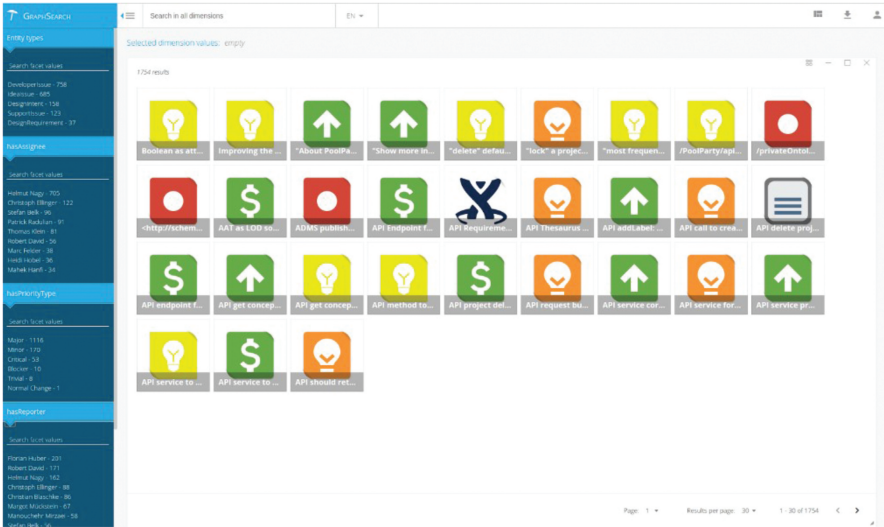


Figure 6.64 Semantic search over development artefact – Graph Search.

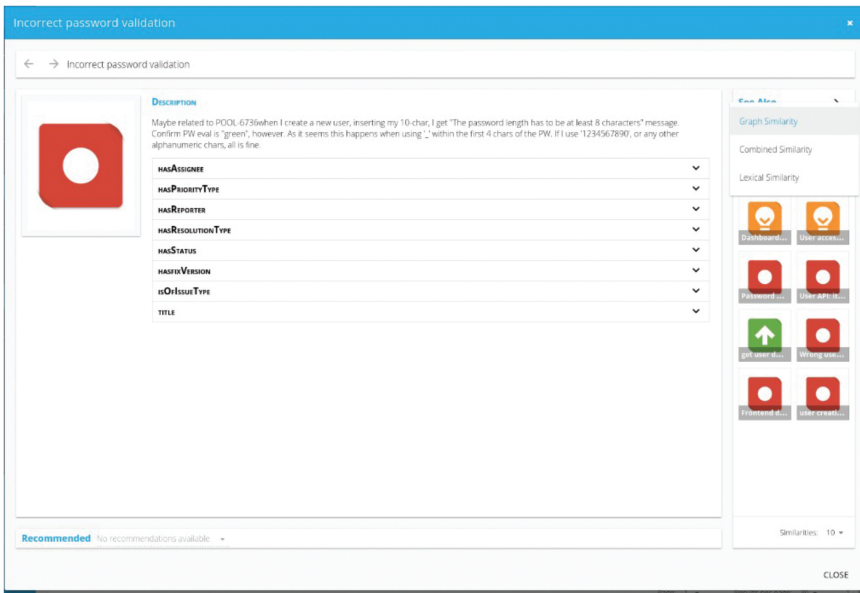


Figure 6.65 Details view of specific issue with the option to select similarity algorithm – PP Recommender.

ALIGNED provides identification and repair of data problems to customers. Curation reduces the need for consultant assistance, but also makes PoolParty more flexible regarding stability by allowing more freedom on data imports. This should raise the acceptance of the product, but it is hard to quantify. Value for the development process can be measured by the reduction of time efforts.

6.4.6.2 Data quality dimensions and thresholds

Data quality requirements for PoolParty are covered by two classes of validation. First, the Import Assistant feature ensures that imported data do not conflict or violate the functionality of the application and therefore provides a stability guarantee for users regardless of the data they import. Second, quality checks can be applied to the data to discover design flaws in the data modelling. These do not cause problems for the application, but might be unwanted by the data engineers. In ALIGNED, we focus on the Import Assistant and analyse the time saved by automatic identification and repair done by the user in contrast to a manual repair by a SWC consultant done directly on the data. Value is represented first by providing a feature for the user to do the repairs without assistance and second by the reduced time efforts needed to resolve data issues.

Data Agility

The Import Assistant provides users with the possibility to safely import any RDF data they want to use. The validation checks are based on stability requirements on the application. They detect all problems that would cause PoolParty to fail, and as a result, any data can be imported safely. Regarding the Unified Governance data, which is also represented as RDF, we can change or extend the set with additional information and adapt the software components using configuration rather than having a need to change the actual implementation. The pipeline processing can be configured within the pipeline steps and the Graph Search faceted search and similarity application uses ontologies to provide both the search interface and the data representation. Also, the Integrated Issue Reporting automatically provides metadata and logging information in the case of application failure.

Model Agility

If new features for PoolParty are implemented or existing features are changed, there might be the need for additional or modified validation checks.

Using a standard-based declarative approach for these allows development to add, change, test and reuse them more easily. The starting point was SPARQL-based checks, and we moved on to using SHACL shapes for validating the import data. The Unified Governance data model can be easily configured using ontologies and therefore provides adaption for the application to a changed dataset. In addition, Graph Search provides a plugin architecture to add new functionality regarding similarity and recommendations. Plugins are automatically loaded and provided via user interface for users to work with.

For PoolParty, the evaluation will compare productivity, usability and data quality, as well the connection between data development life cycle and the software development life cycle to the results in the previous validation deliveries. SWC is evaluating the PoolParty trial for import validation by using the Import Assistant to ensure data consistency. Improvements to the software developing process by using the Unified Governance methodology and tool chain, including the Integrated Issue Reporting, are evaluated regarding time efforts.

6.4.6.3 Evaluation tasks

Curation

Import data into a PoolParty project, detect the problems and repair them.

Process

- Import an RDF dataset into a PoolParty project
- Test the PoolParty Thesaurus Manager to detect problems
- Problems may show up immediately or when a specific resource is addressed
- Repair the problems manually using SPARQL

Challenges

Detection of problems is difficult. It may be that they are discovered during later work. There is no systematic checking other than manual testing, which is a lot of effort and not possible for big Thesauri.

Detection, but especially repair, requires detailed knowledge of the data model and also the PoolParty application. It is unlikely that the average user has this knowledge and can fix the issues.

Data Agility

Use the development data for managing the development process by organising issues to detect duplicates and find similar issues so that requirements and stories can be viewed in relation.

Process

- Duplicates are often created when two different users use a new or modified feature and report a bug. We can only detect duplicates manually at a later time.
- Requirements are manually linked to Jira issues on creation.
- Search has to be done separately in Confluence and Jira.
- In the case of an application failure, a Jira issue is created manually. Log files have to be requested from the customer.

Challenges

- Duplicates are created and detected at a later time. We cannot prevent the duplicate reporting.
- Finding stories and bugs for a specific requirement is a lot of effort. Using linking between Confluence and Jira is done manually.
- We cannot do an integrated search over the whole development data.
- Requesting log files from the customer in the case of application failure increases the time until issues are solved.

Model Agility

Create a unified view on the complete development data.

Process

Development data are managed in separate applications without integration using a common basis.

Challenges

- The development data are distributed in several systems.
- There is no integrated semantic description for the different parts.
- The representation is not standards based. Publishing is difficult.
- Integration and processing is proprietary.

- There is no option to change the processing using a declarative approach like ontologies.

6.5 Data Validation at DBpedia

6.5.1 Introduction

DBpedia is the centre of the current web of data. It publishes authoritative RDF-based datasets that are used as a common point of reference for interlinking and enriching most of the structured data on the Web today. It relies on an automated data extraction framework to generate open RDF data from Wikipedia documents, published in the form of file dumps, Linked Data and SPARQL (SPARQL Protocol and RDF Query Language) hosting on the Linked Data Stack.

DBpedia is a large-scale extraction project of unstructured and semi-structured data from different Wikipedia language editions to RDF. This extraction is achieved from a modular extraction framework that is customised to handle multilingualism and structural differences between different Wikipedia language editions. The latest DBpedia release (v. 2014) generated three billion facts from 125 localised versions. As Wikipedia evolves over time, the code should be able to adapt to these changes. However, identifying errors at this data scale becomes very hard, and validation workflows must be established that will ensure the quality of the extracted data. The high-level goal of ALIGNED in this use case is to produce tools for the DBpedia community, which will increase the coverage and precision of the provided DBpedia data stack.

The latest DBpedia release contains around 23,000 files from more than 100 Wikipedia language editions. At the moment, we provide a download folder for each language and detailed description only for the English dataset. We want to extend the current approach and provide a machine readable representation for the whole release and, besides dataset links to additionally provide descriptions for all datasets and languages, license and contact information using DataID (dataid.dbpedia.org). The DataID will be autogenerated by a script that will iterate over all release folders and using a pattern-based approach will assign metadata for each dataset.

6.5.2 Problem Statement

DBpedia is a large-scale extraction project of unstructured and semi-structured data from different Wikipedia language editions to RDF. This

extraction is achieved from a modular extraction framework that is customised to handle multilinguality and structural differences between different Wikipedia language editions. The latest DBpedia release (v. 2014) generated three billion facts from 125 localised versions. As Wikipedia evolves over time, the code should be able to adapt to these changes. However, identifying errors at this data scale becomes very hard, and validation workflows must be established that will ensure the quality of the extracted data. The high-level goal of ALIGNED in this use case is to produce tools for the DBpedia community, which will increase the coverage and precision of the provided DBpedia data stack.

6.5.2.1 Actors

<i>Role</i>	<i>Description</i>
<i>Extractors</i>	DBpedia team members who run the extraction process for a given DBpedia release
<i>Extraction Agents</i>	Software agents that perform the extraction such as DBpedia live
<i>Mapping editors</i>	Community members who edit the DBpedia mapping wiki
<i>Ontology Editors</i>	DBpedia foundation members that edit the DBpedia ontology
<i>Release managers</i>	DBpedia team members that are responsible for the actions leading to a given release of DBpedia
<i>Developers</i>	DBpedia team members who write code for the extraction tools
<i>Users</i>	Users of DBpedia

The requirements on which the DBpedia use case was based are detailed in Appendix A.

6.5.3 Architecture

Figure 6.66 depicts the DBpedia use case trial architecture, showing the ALIGNED tools used in different stages of the DBpedia data workflow. With the DBpedia trial, we want to showcase both the reuse of ALIGNED tools as well as different integration points. For the data validation trial, we focus on validating instance data, mappings to RDF and links to other datasets. Link validation is performed with the SUMMR Mapping tool that reports results in the ALIGNED Metamodel version 2, especially the DLO and the DBpedia use case specific ontology. This means that the RDF logs produced

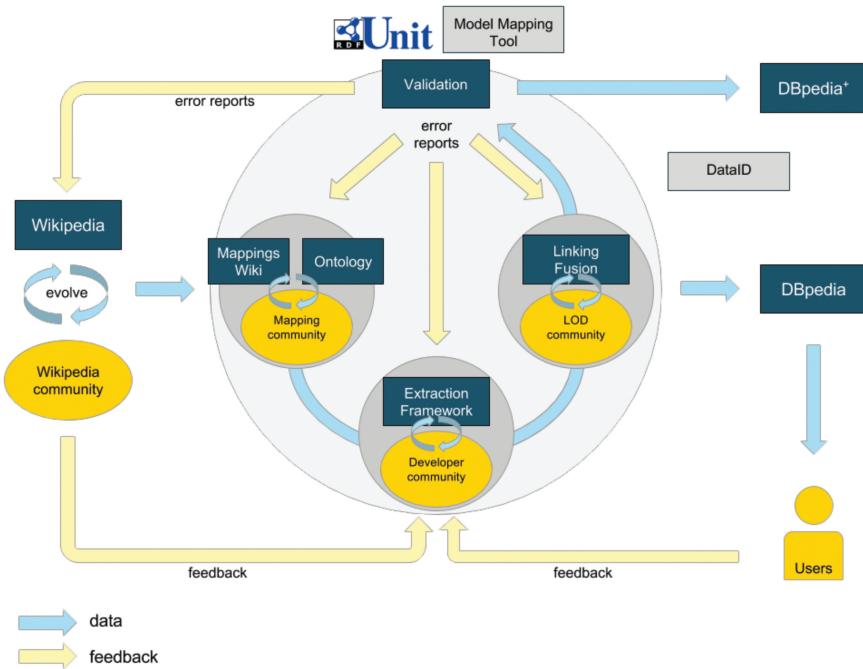


Figure 6.66 DBpedia Use Case Trial System Architecture, showing the ALIGNED tools used in different stages of the DBpedia data workflow.

by SUMMR can be consumed by the ALIGNED Unified Governance Tools in an ALIGNED tool chain. The other two validation scenarios are based on RDFUnit and use the SHACL violation reporting vocabulary as an integration point. DBpedia instance data are validated with a simple RDFUnit setup, while the DBpedia infobox-to-ontology mappings by using RML as an intermediate format. Regarding data dissemination, we use the DataID ontology as an integration point and automate the generation of the download page or a DBpedia release as well as the provision of a DBpedia release in a triple store through Docker.

6.5.4 Tools and Features

Figure 6.67 shows which features of the ALIGNED software tools are deployed in order to support these scenarios.

Software	Feature	Used For
RDFUnit	Data validation	Instance data validation
RDFUnit	Data validation	Mapping Validation
RDFUnit	Data validation	Ontology Validation
RDFUnit	Data validation	Link Validation (for metadata)
SUMMR Mapping Tool	Link Validation	DBpedia Interlink validation
DataID	Dataset description	Automatic generation of a release download page
DataID	Dataset description	Automatic generation of a triple store with data from a release using docker

Figure 6.67 ALIGNED Tools and Features used in the DBpedia trial platform.

6.5.5 Implementation

Figure 6.68 depicts an RDFUnit validation report of a DBpedia release. The report is provided as an RDF file that adheres to the RUT ontology, as well as an HTML export that is human readable. The report provides a high-level overview at the top with basic provenance metadata and statistics and continues with detailed error counts per constraint.

The mapping validation report (Figure 6.69) uses RDFUnit in the background but performs more sophisticated validation processing and reporting. The complete workflow is described in “Assessing and Refining Mappings to RDF to Improve Dataset Quality”. The end user report is tailored for the mapping editors where they can select mappings errors based on language, infobox, DBpedia property, or DBpedia class.

```

TestExecution: http://rdfunit.aksw.org/data/results#ecd98404-2ae1-11b2-8016-d4ae52e56c05
  Dataset      http://dbpedia.org
  Test suite   http://rdfunit.aksw.org/data/testsuite#10ac4098-2ae2-11b2-8016-d4ae52e56c05
  Test execution started 2015-07-06T08:43:31.099Z
  -ended          2015-07-07T01:34:45.948Z
  Total test cases 9468
  Succeeded       7767
  Failed         1701
  Timeout / Error T:0 / E: 0
  Violation instances 12014757

```

Results

Status	Level	Test Case	Errors	Prevalence
Fail	ERROR	http://dbpedia.org/ontology/apparentMagnitude has rdfs:domain different from: http://dbpedia.org/ontology/Planet	177	177
Fail	ERROR	http://dbpedia.org/ontology/geneLocation has different range from: http://dbpedia.org/ontology/GeneLocation	56	56
Fail	ERROR	http://dbpedia.org/ontology/capital has rdfs:domain different from: http://dbpedia.org/ontology/PopulatedPlace	1447	1448
Fail	ERROR	http://dbpedia.org/ontology/internationalAffiliation has rdfs:domain different from: http://dbpedia.org/ontology/PoliticalParty	757	757
Fail	ERROR	http://dbpedia.org/ontology/distanceToCharingCross has rdfs:domain different from: http://dbpedia.org/ontology/Settlement	161	161
Fail	ERROR	http://dbpedia.org/ontology/mountainRange has rdfs:domain different from: http://dbpedia.org/ontology/Mountain	10449	10450
Fail	ERROR	Two different dbo:Country(s) should not have the same dbo:capital	417	934

Figure 6.68 Instance data validation report with RDFUnit.

Show entries

Search:

language	mapping	predicate	expected	existing
en	<input type="text" value="mapping"/>	<input type="text" value="of"/>	<input type="text" value="expected"/>	<input type="text" value="existing"/>
en	Infobox_sea (info)	areaOfCatchment (info)	Lake (info history)	Sea (info history)
en	Infobox_rail (info)	voltageOfElectrification (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Infobox_rail (info)	typeOfElectrification (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Infobox_public_transit (info)	voltageOfElectrification (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Infobox_public_transit (info)	typeOfElectrification (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Infobox_public_transit (info)	numberOfStations (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Football_box (info)	numberOfGoals (info)	CareerStation (info history)	Event (info history)
en	Infobox_Geopolitical_organization (info)	officialLanguage (info)	PopulatedPlace (info history)	Organisation (info history)
en	Infobox_train (info)	yearOfConstruction (info)	Place (info history)	Train (info history)
en	Infobox_television_season (info)	numberOfEpisodes (info)	TelevisionShow (info history)	TelevisionSeason (info history)
en	Infobox_diocese (info)	numberOfMembers (info)	Legislature (info history)	Diocese (info history)
en	Infobox_canal (info)	riverBranchOf (info)	River (info history)	Canal (info history)
en	Infobox_bus_transit (info)	numberOfStations (info)	RouteOfTransportation (info history)	PublicTransitSystem (info history)
en	Infobox_bay (info)	areaOfCatchment (info)	Lake (info history)	Bay (info history)
en	Infobox_academic_conference (info)	frequencyOfPublication (info)	PeriodicalLiterature (info history)	AcademicConference (info history)

Showing 1 to 15 of 15 entries (filtered from 2,292 total entries)

Previous Next

Figure 6.69 Mapping validation report with RDFUnit and RML.

Starting in October 2017, DBpedia replaced the old Mappings Wiki with a new Mappings UI, based on GitHub and RML mappings. The validation report shown in Figure 6.70 has been superseded by this interface, since wrong mappings are detected automatically on commit.

The graphical interface based on the DBpedia-Links repository provides an overview of all outgoing links to other datasets and points out any inconsistency in a given linkset (Figure 6.71). DBpedia employs multiple validation methods for link validation, including the SUMMR Mapping tool (see below).

The SUMMR Mapping tool (Figure 6.72) performs an interlinking validation for all the external links in a DBpedia release. After processing the link, the tool outputs a log file and splits the links into valid and invalid. The invalid links are discarded, and only the valid ones become part of the DBpedia release.

Gathering extensive metadata throughout all extraction steps is not only helping to produce exhaustive dataset metadata (in form of DataID documents), but also allows for highly expressive logs and convenient summary reports (as shown in Figure 6.73).

The generation of a DBpedia release download page was a tedious task. We use DataID as a core release metadata component and created a flexible user interface that people can use to identify and filter specific DBpedia datasets. Figure 6.74 depicts the download page of the 2015-2010 release.

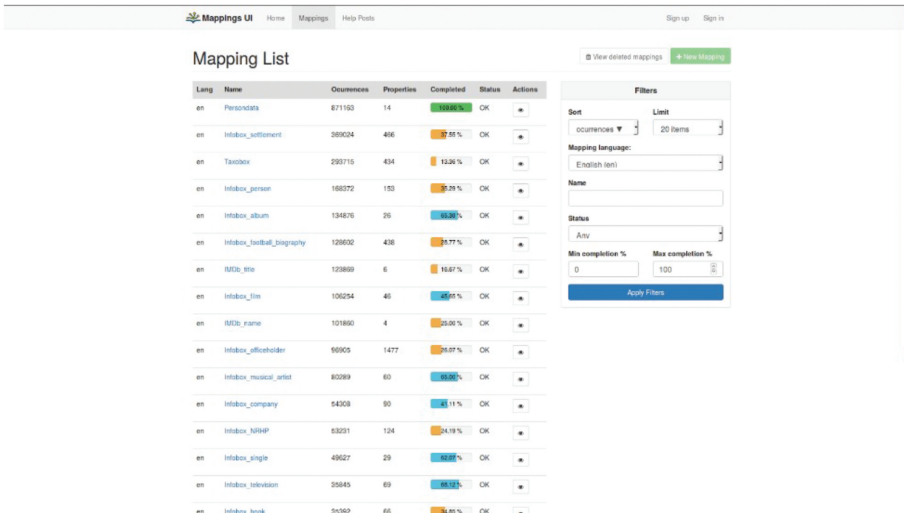


Figure 6.70 The new Mappings UI (using RDFUnit for validating mappings).

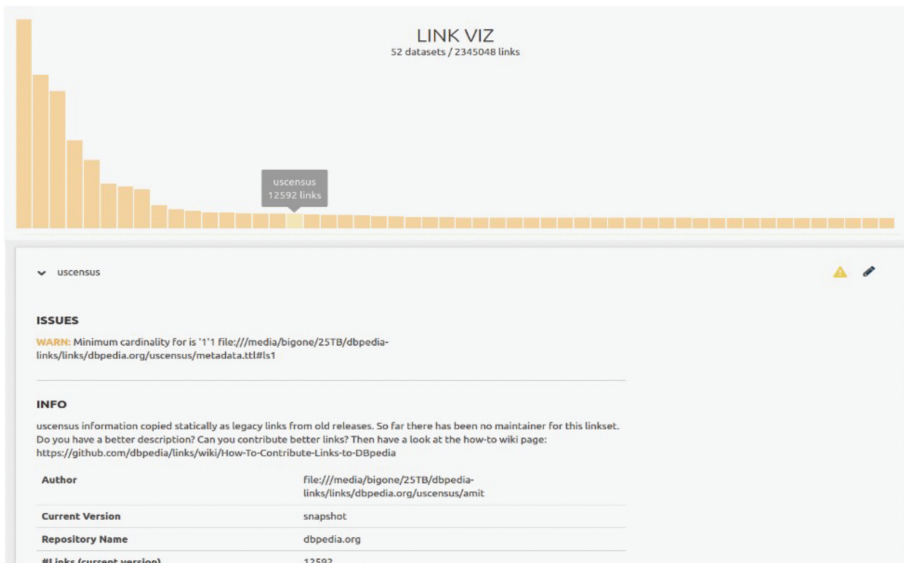


Figure 6.71 DBpedia Link Viz tool.

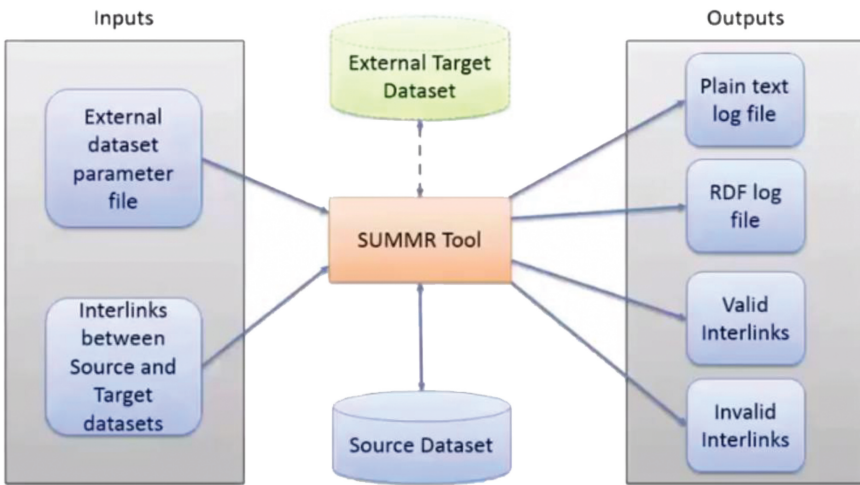


Figure 6.72 SUMMR Mapping tool.

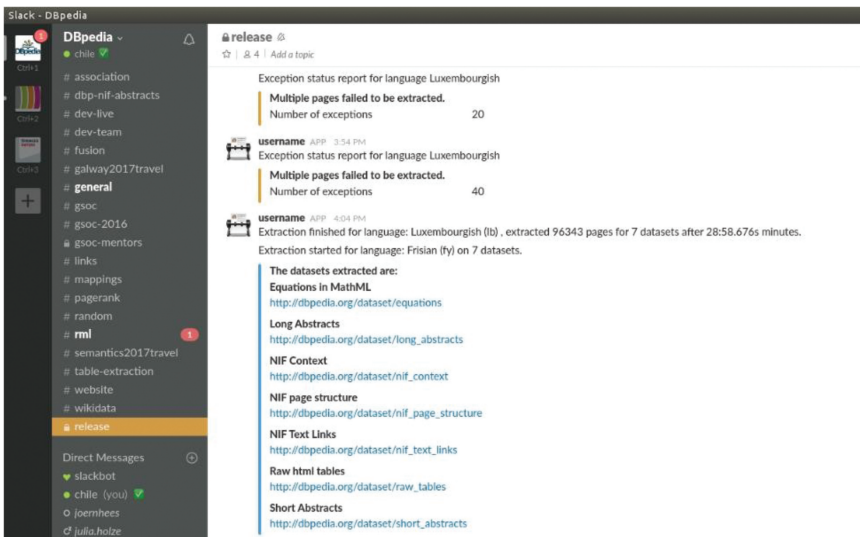


Figure 6.73 Active extraction monitoring (here: extraction summaries forwarded to Slack).

The dockerised DBpedia (Figure 6.75) automates the digestion of a DBpedia release by downloading the datasets of the user’s preferred language and loading these datasets on a Virtuoso triple store server. We use docker as the underlying technology that has recently become a very common means

de x en x es x fr x ja x nl x pt x ru x show languages

Search:

Dataset	en	de	es	fr	ja	nl	pt	ru
geo coordinates	tq1 ?	tq1 ?	tq1 ?	tq1 ?	tq1 ?	tq1 ?	tq1 ?	tq1 ?
mappingbased	t11 ?	t11 ?	t11 ?	t11 ?	t11 ?	t11 ?	t11 ?	t11 ?
mappingbased literals	tq1 ? t11 ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?
mappingbased objects	tq1 ? t11 ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?
mappingbased objects disjoint domain	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?
mappingbased objects disjoint range	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?	tq1 ? t11 ?
specific mappingbased properties	tq1 ? t11 ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?	tq1 ? tq1* ? t11 ? t11* ?

Figure 6.74 DBpedia download page through DataID.

dbpedia / Dockerized-DBpedia Unwatch 9 Unstar 11 Fork 3

[Code](#) [Issues 3](#) [Pull requests 0](#) [Wiki](#) [Pulse](#) [Graphs](#) [Settings](#)

creates a docker image with Virtuoso preloaded with the latest DBpedia dataset – Edit

28 commits | 1 branch | 0 releases | 3 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

- neradis minor: delete misplaced space Latest commit b25b4c9 on Apr 18
- docker increased store initialisation timeout (in case of slow or busy host ... 6 months ago
- .gitignore added .gitignore 6 months ago
- Dockerfile Removing figlet as a dependency 4 months ago
- README.md minor: delete misplaced space 4 months ago
- download.sh Use core as a language option 4 months ago

Figure 6.75 Dockerised DBpedia.

of application distribution. DataID is used as the core metadata component to identify and filter DBpedia datasets.

6.5.6 Evaluation

6.5.6.1 Productivity

The basic unit of analysis for productivity is a comparison of time elapsing between two DBpedia releases. Typical tasks are code maintenance, release

management, ontology editing, release documentation creation and dealing with user queries.

For a sufficient evaluation of productivity changes between two DBpedia releases, one has to consider the changes to data sources, ontology, mappings, and the code base. In addition, the number of published datasets tends to increase over time when incorporating new extraction methods and algorithms. Nonetheless, over the time of this project, DBpedia has managed to cut the time between releases in half (13 months to 6 months), while producing at least three times as many pieces of information (triples). Currently, DBpedia is pushing for regular updates for the 10 most widely used language editions on a bi-monthly basis (synchronising with the bi-monthly data releases by Wikimedia). Multiple efforts to increase productivity are closely related to Quality and Agility (see below).

A significant improvement of time spent on dissemination activities was achieved by introducing DataID as dataset metadata format. Extensive metadata descriptions of datasets allow for many automation tasks, such as automated downloading of relevant dataset files, generic implementations of dataset overviews, and download tables. In addition, extensive and high-quality metadata of datasets helped DBpedia to check 31 of the 35 Data on the Web Best Practices of the eponymous W3C working group.

6.5.6.2 Quality

Instance Validation

To create high-quality data, a validation method for DBpedia instance data has to provide sufficient metadata to distinguish between three different possible sources of a violation:

- The Wikipedia editor (entering erroneous values)
- Incorrect mappings, between source and DBpedia ontology
- A software issue in the DBpedia Extraction Framework

RDFUnit was created with these demands in mind, providing necessary metadata to any violation found and creating links between a software issue and the violating instance (see D5.8). The resulting violations and their pertaining metadata provide the exact coordinates of a violation, the grounds for this violation and the possible source. Thus, violations recorded in such a manner are used as feedback medium, relating possible mistakes to Wikipedia editors, the mapping community or software developers. DBpedia is running all published data through RDFUnit, validating it against an up-to-date version of the DBpedia ontology. The validated outputs generate consistent data that

are termed DBpedia+, whereas the wider, more exhaustive data are published as the standard DBpedia datasets.

Mapping Validation

In addition to validating the resulting instance data, DBpedia started to validate the mappings between DBpedia ontology and the Wikimedia data sources on a nightly basis with RDFUnit. Thus, most of the mapping-related violations can be caught before ever starting the data extraction, preventing possible reruns of whole extraction steps and increasing productivity in turn.

Ontology Validation

The DBpedia ontology has been maintained by the DBpedia community in a crowdsourced manner at the mappings wiki. There is an ongoing effort to move ontology development onto GitHub for easier collaboration and for the sake of more control over the ontology structure.

At the time of writing, a set of constraints ensure that each DBpedia class and each DBpedia property conform to DBpedia community requirements. RDFUnit is used to perform the validation (using SHACL constraints) and to integrate with Travis CI and automate the checks on each commit and pull request.

Link Validation

The DBpedia-Links repository maintains linksets between DBpedia and other LOD datasets. A system for maintenance, update and quality checks, which validates various aspects of the link submission, is in place and is integrated with common continuous integration services, such as Travis CI. It offers a way to publish linksets between DBpedia and any given dataset, which are published alongside the DBpedia dataset files.

Quality checks include:

- The SUMMR Mapping validation tool
- RDFUnit for validating (using SHACL constraints) the link manifest (basic metadata providing a minimum of provenance)

Workflow Validation

To ensure quality regarding the extraction workflow, DBpedia extended the extraction framework to produce metadata for any extraction process, extensive logging of progress and exceptions, as well as high-level summaries

of extractions. These efforts support extensive monitoring, metadata propagation and logging (on triple and dataset level) and the deployment of ETL frameworks and Workflow Management Systems to further decrease the time needed for extraction and to automate this process completely. Currently, a concerted effort to adapt the Unified Views Framework of SWC for this purpose is underway, which will continue until after this project has finished.

6.5.6.3 Agility

The greatest need for agility in DBpedia is the ability to rapidly respond to changes in source datasets like Wikipedia. This is the focus of the use case scenario Wikipedia/Wikidata change. Example Wikipedia changes that impact DBpedia are: the introduction of new pages that represent new concepts, the introduction of new infobox templates that represent additional instance data in DBpedia, and changes in infobox structures. Adapting to those changes in a (semi-) automated way will prevent the loss of data (due to changes to Wikipedia templates) and incorporate new instance data automatically.

As a prerequisite to automate mappings, DBpedia will switch its complete mapping infrastructure to an RML-based mapping approach in October 2017. This is a direct result of one of our Google Summer of Code projects of 2016. As a superset of the W3C recommended mapping language R2RML for relational databases to RDF, RML offers a way to completely represent all DBpedia mappings in RDF. It enables:

- Full support of RDFUnit mapping validation (no transformation necessary)
- The complete range of mapping possibilities of RML (incl. functions, conditions, etc.)
- Rule-based automation of mappings using all RML features
- Replacing the rigid wiki text mappings used by DBpedia until now

Concurrently, DBpedia helped to implement a taxonomy learning system based on Wikipedia categories. Set up as one of our annual participation with the Google Summer of Code program, this project realised the concept laid out in the “Unsupervised Learning of an Extensive and Usable Taxonomy for DBpedia”. These automatically derived types are a reliable backbone for the automated mapping generation ahead.