2

ALIGNED Use Cases – Data and Software Engineering Challenges

Arkadiusz Marciniak and Patrycja Filipowicz

Adam Mickiewicz University, Poland

2.1 Introduction

The ALIGNED project developed an aligned methodology for parallel software and data engineering of Web-scale information systems with Linked Data as a unifying technical foundation for system specification and process and tool integration. This methodology (see Chapter 3) is based on a metamodel describing the complete software and data life cycles, domain models, and design intentions. This metamodel specifies tools to produce software development models, including transformations that generate or configure software applications as well as data development models, incorporating data quality and integrity constraints, data curation workflows, and data transformations.

Software and data engineering are different disciplines, with different practices and processes. Significant differences between these fields mean that a single prescriptive approach could not work. Instead, the project has identified a matrix of synchronisation points between different stages of the software and data life cycles. Each point represents a key area where software and data engineers may need to interact and define formats and processes for working together. This approach is flexible enough to accommodate many different workflows, while still identifying key areas where alignment of the two life cycles can lead to significant savings in effort. The approach adopted endeavoured to make it possible to improve the overall quality, productivity, and agility in a variety of different use cases. In order to achieve these objectives, the project sought to develop Linked Data schemata for alignment that enabled the software engineering life cycle of data-intensive systems to be integrated with the data engineering life cycle, by identifying common

phases and signalling between the parallel processes and tools to support alignment at higher levels.

The decision to adopt data-model-driven approaches in the project had far-reaching consequences. In particular, it required that every step in the process be directly driven by the model, rather than independently configured. Harvested datatypes also could not be consumed directly, but through a model, which dictated the shape and structure that the data must take. Accordingly, a model-driven approach led to the creation of explicit models at each stage of the development process.

MDE describes a development process in which the components of the final software artefact are derived – either manually or automatically – from models that typically form part or all the specifications or requirements of the system. The software needs to be written in such a way that it understands the modelling language and is capable of handling updates to the model. Such software can be reused in different applications within a similar domain, minimising the time spent on the implementation phase, and capturing common repeating patterns that would otherwise have to be repeated in each cycle of an iterative development. In the MDE world, it is required that the data model is provided in the form of the ontologies available at a well-known URL, which is typically achieved by providing a metadata registry.

In order to achieve the postulated goals, a number of tools from both domains were developed and used in order to make the advocated integration of both life cycles efficient, particularly in relation to challenges posed by the different use cases. These comprise Booster, the Model Catalogue, RDFUnit, Repair Framework and Notification (RF), Ontology Repair and Enrichment (ORE), Dacura, the PoolParty Confluence/JIRA Data Extractor (CJDE), External Link Validation (ELV), and the Unified Governance Plugins.[1] Similarly, a set of open, public ontologies and vocabularies were adopted and used wherever possible by all tools to support integration (for details, see Chapter 4). These include foundational schemas, such as RDF, RDFS, and OWL, and common widely used standards such as PROV and SKOS. Where ontologies did not exist to cover the advocated integration needs, new models were created and made publicly available (RVO, RUT, DataID). This collection of common, project-wide ontologies gave the ability

[1]Shah, Seyyed M., James Welsh, Jim Davies, and Jeremy Gibbons. 2017. In Mahmood, Z (ed.), Software Project Management for Distributed Computing: Life-Cycle Methods for Developing Scalable and Reliable Tools, 367–385. Springer: Cham.

to exchange rich, structured information covering the most significant entities within research focus.

The ultimate objective of the project, however, was to produce tools, methods, and standards, which lead to real improvements in productivity, quality, and agility of different types of data. The rapidly increasing size and complexity of Web and Big Data often makes their management virtually impossible, where even specialists struggle to harness them. Hence, five use cases representing different domains from legal to health and complex archaeological and historical datasets were chosen to adopt a broad bottom-up approach to system development and integration. Accordingly, the project tackled problems in a wide range of areas with the intention to show how the latest semantic technologies can help create means of managing and using these datasets. The selection of uses cases was also driven by a need of testing interoperability between the tools, particularly those who support both data and software engineering that were developed in the project. The chosen use cases were: (i) Seshat: Global History Databank, (ii) PoolParty Enterprise Application Demonstrator System, (iii) DBpedia, (iv) Jurion and Jurion IPG, and (v) Health Data Management.

Each use case is a large-scale, real-world project with large user communities and complex sets of data. The project's research has thus had a practical focus, which has seen the application of innovative tools and solutions in real life. The use cases represent diverse domains, both commercial and non-commercial, which have their own requirements and data characteristics. They also represent a significantly different level of advancements in both the data and software engineering tools and procedures. Each use case has its own problems with quality, agility, and productivity. The project has built tools and processes that improve software and data engineering for each of these use cases. Every tool appears in more than one use case, and every use case involves tools developed by different partners. In each case, trial platforms were constructed in multiple phases, which integrate research outputs from multiple work packages and partners, served to offer the greatest potential for real improvements to the existing processes employed within these use cases.

The objectives of this chapter are thus threefold: (1) to present the five case studies used in the ALIGNED project, (2) to analyse the major challenges identified by these use cases in the data engineering life cycle as well as present their proposed solutions, and (3) to analyse the major challenges identified by the use cases in the software life cycle and propose solutions to these challenges.

2.2 The ALIGNED Use Cases

2.2.1 Seshat: Global History Databank

The Seshat: Global History Databank[2] is an international initiative of humanities and social science scholars to build an open repository of expert-curated historical time-series data.[3] The Seshat project began by selecting a sample of 30 areas from around the world. For each area, all societies that had controlled it throughout history were recorded. 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.[4] The databank extracts data from a combination of databases, Linked Data sources, websites, academic publications, and human experts. Figure 2.1 shows the initial sample of 30 geographical areas chosen for the databank.

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,

[2] http://seshatdatabank.info

[3] Turchin, Peter, Thomas E. Currie, Kevin C. Feeney, Pieter Franois, Daniel Hoyer, J.G. Manning, Arkadiusz Marciniak, Daniel Mullins, Alessio Palmisano, Peter Peregrine, Edward A.L. Turner and Harvey Whitehouse Harvey. Seshat, The Global History Databank, Cliodynamics. The Journal of Quantitative History and Cultural Evolution 6(1), pp. 77–107.

[4] Turchin, Peter, Thomas E. Currie, Harvey Whitehouse, Pieter Franois, Kevin Feeney, Daniel Mullins, Daniel Hoyer, Christina Collins, Stephanie Grohmann, Patrick Savage, Gavin Mendel-Gleason, Edward Turner, Agathe Dupeyron, Enrico Cioni, Jenny Reddish, Jill Levine, Greine Jordan, Eva Brandl, Alice Williams, Rudolf Cesaretti, Marta Krueger, Alessandro Ceccarelli, Joe Figliulo-Rosswurm, Po-Ju Tuan, Peter Peregrine, Arkadiusz Marciniak, Johannes Preiser-Kapeller, Nikolay Kradin, Andrey Korotayev, Alessio Palmisano, David Baker, Julye Bidmead, Peter Bol, David Christian, Connie Cook, Alan Covey, Gary Feinman, Árni Daníel Júlíusson, Axel Kristinsson, John Miksic, Ruth Mostern, Cameron Petrie, Peter Rudiak-Gould, Barend ter Haar, Vesna Wallace, Victor Mair, Liye Xie, John Baines, Elizabeth Bridges, Joseph Manning, Bruce Lockhart, Amy Bogaard and Charles Spencer. Single dimension of complexity in human societies. Proceedings of the National Academy of Sciences, 115 (2) E144-E151; DOI:10.1073/pnas.1708800115.

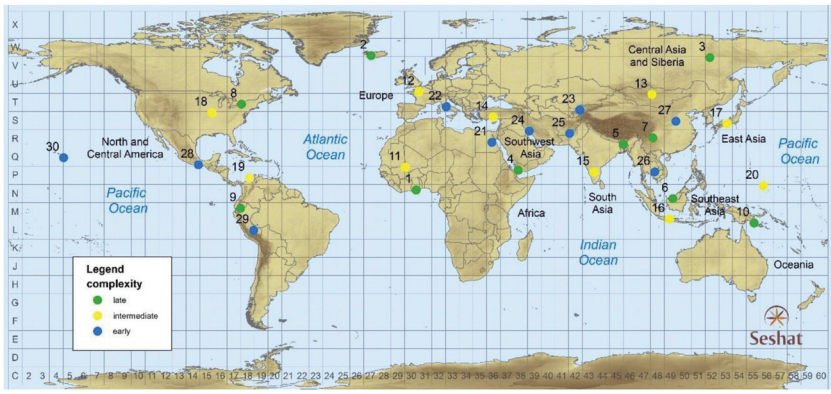

Figure 2.1 Seshat World Sample 30.

it did not scale to the number of contributors, data users, data points, or the complexity of the data.

Seshat also 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, 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. This 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 a separate geographic information system (GIS). The wiki-based system offered no support for publication. Furthermore, while the scraping

tool could extract raw datapoints, important citations and comments 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.

2.2.2 PoolParty Enterprise Application Demonstrator System

The PoolParty Semantic Suite[5] is the SWC's platform for enterprise information integration based on Linked Data principles. The PoolParty semantic technology suite comprises a number of tools based on the extraction, curation, and management of linked open datasets. These tools are split into three categories: data portals and collaboration platforms, tools for knowledge engineering and graph management, and functionality for content enrichment and data integration. Any data is transformed into RDF graphs and can be queried with SPARQL (SPARQL Protocol and RDF Query Language). Since it was created, the product has evolved to include entity extraction from unstructured information. PoolParty's API provides a rich set of methods for text mining and entity extraction. Figure 2.2 shows the tools of the PoolParty Application Suite.

Figure 2.2 PoolParty Application Suite.

[5]http://www.poolparty.biz

As a loosely coupled collection of tools, additional functionality has been enabled through the integration of third-party tools. An example of this is the use of Atlassian Confluence (a team collaboration tool), Atlassian Jira (a tool for issue tracking and project management), and Media Sonar (a Web-mining tool), for a general-purpose requirements engineering system. However, the systems concerned are typically document-oriented and require extensive human interaction in order to link their data to development tasks recorded in PoolParty against standard ontologies. The system lacks the required integration and alignment of data management issues with the software development life cycle, so that each supports the other.

2.2.3 DBpedia

DBpedia[6] 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 hosting on the Linked Data Stack. This structured information resembles an open knowledge graph, which is a kind of database, which stores knowledge in a machine-readable form and provides a means for information to be collected, organised, shared, searched, and utilised. DBpedia passes all published data through RDFUnit, validating it against an up-to-date version of the DBpedia ontology. The validated outputs generate consistent data termed DBpedia+, whereas the wider, more exhaustive data are published as the standard DBpedia datasets.

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: (i) the Wikipedia editor (entering erroneous values), (ii) incorrect mappings between source and DBpedia ontology, and (iii) a software issue in the DBpedia Extraction Framework. Accordingly, RDFUnit provides the necessary metadata for any violation found and creates links between a software issue and the violating instance. The resulting violations and associated 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 a feedback medium, relating possible mistakes to Wikipedia editors, to the mapping community, or to software developers. In addition to validating the resulting instance data, DBpedia started to validate

[6] http://wiki.dbpedia.org

the mappings between DBpedia ontology and the Wikimedia data sources on a regular 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.

The DBpedia Links repository maintains linksets between DBpedia and other LOD datasets. A system for maintenance, updates, 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.

The major productivity issues identified for DBpedia involve code maintenance, release management, ontology editing, release documentation creation, and dealing with user queries. Further complications involved dealing with the increasing number of published datasets that tend to increase over time when incorporating new extraction methods and algorithms.

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 both the 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. Figure 2.3 shows this pipeline.

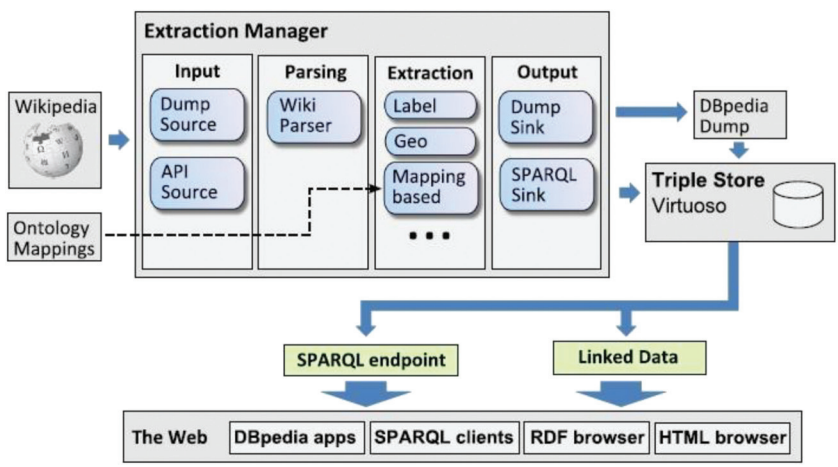

Figure 2.3 DBpedia Extraction Pipeline.

The greatest need for agility in DBpedia is the ability to rapidly respond to changes in source datasets like Wikipedia. These may involve, among others, the introduction of new pages that represent new concepts and 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.

2.2.4 Jurion and Jurion IPG

The Wolters Kluwer[7] use case within ALIGNED is twofold. On the one hand, the project worked with a legal research database application called Jurion (www.jurion.de) from Wolters Kluwer Germany. In this use case, it mainly focussed on addressing data quality issues. Second, the project re-engineered the IPG system from Wolters Kluwer Poland, which is a commercial intelligence system, based on huge amounts of data in a relational database system.

Jurion 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. Wolters Kluwer concentrated mainly on the enhancement of data quality and repair processes. Based on the requirements, it started to work on data transformation issues and the improvement of data quality processes in PoolParty in parallel to some tasks within the PoolParty use case. Based on large amounts of XML data, governed by a DTD, continuous transformation from XML to RDF, based on XSLT scripts, needs to take place. This process is complicated and error-prone, especially when it comes to schema changes. The second major data quality challenge is around domain thesauri and controlled vocabularies. Very often, these data are initially created and stored in XLS files and when it comes to a systematic usage of more powerful tools like PoolParty, the import process of this data needs to be optimised, so that errors and inconsistencies can be detected very early in the process.

[7] www.wolterskluwer.com

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. It has been developed by Wolters Kluwer Poland and it contains data on 450,000 companies, 1.1 million people, and 3.5 million documents. The existing data are currently stored in a relational format. The complexity of the system stems from huge amounts of daily processed data originating from pdf sources and their maintenance through a proprietary, obsolete CMS. In order to remain 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. 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. The major requirements involve deploying new tools to find problems in the existing data, improving the integrity of data submitted in the future as well as help increasing the scope of the data by enabling the linking of data stored within the system to external related datasets. Figure 2.4 shows the JURION IPG workflow.

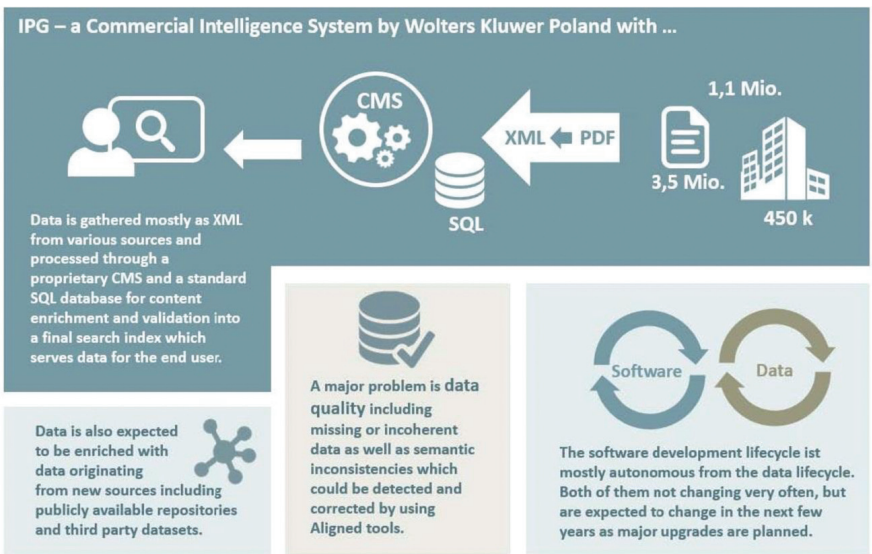

Figure 2.4 Jurion IPG.

2.2.5 Health Data Management

The Health Research Data use case involved four separate projects related to health research data in the United Kingdom:

- the Health Data Finder[8] – an online tool for discovering national healthcare datasets commissioned from the National Institute for Health Research (NIHR). They 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.
- the NIHR Health Informatics Collaborative[9] – routine clinical data in five therapeutic areas provided by the largest teaching and research hospital trusts. These include critical care, ovarian cancer, acute coronary syndromes, hepatitis, and renal transplantation. 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 into account the research purpose.
- the UK 100,000 Genomes Project[10] – a UK Government project aimed at sequencing whole genomes from National Health Service patients. It is focussed on rare diseases, major types of cancer, and infectious diseases. The patients give consent for the genome data to be linked to information about their medical condition and health data. The ultimate goal of the project is to improve knowledge of the causes, treatment, and care of these diseases.
- the construction of a data warehouse for Oxford University Hospitals Foundation Trust[11] – this is 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.

[8] http://www.hdf.nihr.ac.uk

[9] https://www.nihr.ac.uk/about-us/how-we-are-managed/our-structure/infrastructure/health-informatics-collaborative.htm

[10] https://www.genomicsengland.co.uk/the-100000-genomes-project

[11] http://www.ouh.nhs.uk

In all four applications, reuse of existing data without detailed documentation causes major problems, particularly in relation to poorly developed semantics. Furthermore, linkage between datasets may be inaccurate, transformation of data into different formats may be incorrect, and interpretation of statistical results is error-prone. In the Health Data Finder, 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 is often incomplete, incomparable, or completely unusable.

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 and 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.

Furthermore, 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. Implementing efficient database structures requires a lot of repetitive work such as 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.

2.3 The ALIGNED Use Cases and Data Life Cycle. Major Challenges and Offered Solutions

The LOD life cycle consists of eight stages for data engineering.[12] These are: (i) extract – taking information in unstructured form or conforming to other structured or semi-structured formalisms and mapping it to the RDF data model, (ii) storage and querying – retrieving and persisting information in triple form to be included as part of the dataset; (iii) manual revision/ authoring – processes for manual creation, modification, and extension of the structured data; (iv) interlinking/fusion – creating and maintaining links between datasets; (v) classification/enrichment – creating and maintaining links between data, and models of data (which themselves may be linked and part of the dataset); (vi) quality analysis – testing for data completeness and correctness; (vii) evolution/repair – correcting invalid data resulting from a quality analysis phase, via either manual or automated processes; and finally, (viii) search/browse/exploration – making data artefacts available to domain experts or to users beyond the original authors.

Different stages of data engineering in the ALIGNED project have been identified primarily for building tool support and integrated frameworks as well as encouraging compatibility of independent tools within a particular framework. Feedback from one phase is to be fed into another. For example, the models linked during the classification or enrichment stage will determine the scope of the quality analysis stage, or any errors found during quality analysis may need to be resolved in the evolution/repair phase.

As the dataset grew and the focus moved from collection to analysis, several significant problems with agility, quality, and productivity emerged. First, the fundamental problem was that a wiki is designed for human presentation and not machine-readable. Second, the limitations of the wiki impacted agility: manual data harvesting has been very time-consuming. Finally, productivity suffered as increasing resources had to be devoted to curation and cleaning.

In each use case, ALIGNED technologies are being used in slightly different ways. In case of Seshat, these tools are automatically generated from the Seshat ontology. These comprise the Model Mapping Tool, Real-time Instance Data Validation, and curation workflows, all deployed as Dacura services. Dacura is a data curation platform developed by Trinity

[12] Shah et al. 2017: 370.

College Dublin, which incorporates several techniques. The adopted solutions improved the process of data collection.

The model catalogue tool is used in the analysis phase of model-driven software engineering to explore and gather metadata related to the system under construction. It is also used in its search browse and phase life cycle. In the project, it is primarily the Model Catalogue that is used along with components of Semantic Booster, both developed by Software Engineering at Oxford University. In the data engineering context, tools generated by Booster can be used to provide a well-defined API as well as to search and gather data into the data store. Booster-generated systems provide, create, read, update, and delete functionality for data in a data store, as well as implement any user-specified action, which can then be accessed as triples via an API (Shah et al. 2017: 381).

In the Jurion use case, an enhanced data quality and repair pipeline was established with the help of RDFUnit and PoolParty, so that data life cycle process was suffering from less data errors and schema inconsistencies and the overall process was accelerated, especially when data or schema changed over time.

In the Jurion IPG use case, the Model Catalogue is 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 data within the organisation. The Model Catalogue also aims at serving as a provider of models to the generated tools and as an environment where new versions of the data model can be created and evolved. Dacura was instantiated as an alternative approach, covering the overall process from model storing, mapping, and a complete automatic generation of the final future data schema, accompanied by automatic data testing with RDFUnit.

In the NIHR Health Data Finder, the Model Catalogue is the central resource, holding the master copy of models and documentation. In the NIHR Health Informatics Collaborative, each site hosts its own instance of the Model Catalogue, documenting their own data landscape including 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 schema 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 (extract, transform, load) process from OpenClinica, 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. 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 is to be used in the construction of research data extracts and for generating hospital auditing and service improvement metrics.

One of the major steps in data engineering life cycle was the development of new approaches to data validation. In particular, it comprised a new tool developed for the PoolParty semantic suite. The process involved importing RDF data in PoolParty and using the integrated validation checks to identify problems, which are reported to the user as constraint checks. The user is then given options to repair the data consistency. After fixing the inconsistencies, the user can then import the data without the risk of application failure.

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. They together formed the basis for the Data Quality Framework and the Automated Data Testing and Verification Framework.

In most cases, constraint violations only become apparent after the import has been done. In the worst case, this may even cause issues displaying the data or errors displayed to the user. In other cases, issues could pass through unnoticed or may only become apparent at a later stage. This means that users interpret data issues as software issues and report those as bugs in the SWC support space. Import validation has the potential to provide major improvements of productivity and data quality in the data development life cycle. The prototype import validation implemented in PoolParty using RDFUnit enabled the users to get direct user feedback on violations of data constraints. The feature provides direct feedback on data consistency constraint violation before data are imported. Being able to detect violations of consistency constraints on data import increases data quality, since problems are not imported into the system in the first place. The import validation

features provide increased agility, empowering users to import data without quality issues. That means, users can react to issues themselves and fix the data before it gets imported. This improves the connection between the data development and the software development life cycle.

Notification can improve the usability of the PoolParty software, actively providing notifications to users based on activity in projects. Currently, staying informed about activities in projects can only be achieved by reviewing the project history regularly. The ORE tool suggests new ontology axioms (enrichment) and recommends semi-automatic fixes (for resolving violations).

Another important contribution of the ALIGNED project was in the domain of search, browsing, and exploration. Of particular importance is Dacura, which is in a position to produce data quality tolerance requirements to constrain the data to be harvested.[13] The CJDE tool is also responsible for extracting relevant requirement information and hence, tickets and creates RDF data.

The Dacura approval queue allows also dataset administrators to monitor added data for quality and completeness. Administrators can approve, deny, publish, and unpublish the Linked Data objects submitted by Seshat researchers. From a Dacura point of view, it is possible to import large volumes of IPG data into a structured, rich semantic format according to a predefined model that is amenable to statistical analysis and offers automated quality control. Dacura ensures consistency requirements and allows users to monitor newly added data with respect to quality and completeness conditions according to defined constraints.

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

2.4 The ALIGNED Use Cases and Software Life Cycle. Major Challenges and Offered Solutions

The LOD life cycle consists of five stages for software engineering, including: (i) planning – assessment of the feasibility of software to fulfil the requirements of the user; (ii) analysis – identifying potential problems;

[13] Shah et al. 2017: 381.

(iii) design – specification of software intended to achieve the specified goals, including recognition of necessary components and existing constraints; (iv) implementation – installation of the software on user machines; and finally, (v) maintenance – controlling and checking the performance of the software.

The simplest form of the software development is the waterfall model. It is assumed that each element in the life cycle is completed in an unproblematic fashion and there is no need to refer to the previous stage in implementing the process. However, the major problem with this model is that the execution of one phase of design may influence the previous stage. This is particularly apparent in the verification stage when issues in implementation and verification will require further effort in design, which means that design may be said to be unfinished until verification is complete. This may also hold true in the case of planning and specification while the process of producing a clear, precise specification may uncover ambiguities or inconsistencies in the requirements provided.

The integration of both life cycles is only possible when the data engineering systems, such as Dacura, provide 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 traditional Web developers. For example, a GeoJSON stream is automatically made available describing all the features in the dataset that have a geographical location associated with them.

The data model developed by the Semantic Web community was made available to software engineers by providing a metadata registry. The Model Catalogue discussed above is such a registry. It can also be defined as a toolkit for creating and managing data models. 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 incorporation into complex workflows. It was also integrated into the Eclipse Modelling Framework, allowing existing tools to more easily use the catalogue for development. Plugin capabilities were added, facilitating the extension of the catalogue to allow it to interact with more data sources. Semantic reasoning and search were also added, allowing the more efficient reuse of ontologies and concepts. The Model Catalogue was used for Seshat, Jurion IPG and Health Data use cases. In case of Seshat, the Model Catalogue tool allowed the creation of complex ontology to capture the complex historical data the project is collecting.

The Unified Views tool is an ETL tool for RDF data developed as part of the PoolParty semantic suite. It was used to manage the integration of datasets from third-party datasets. The development artefacts are imported into the triple store using a UnifiedViews pipeline. This pipeline runs daily to keep the data up to the date. The pipeline also calculates similarities between the issues and requirements. This solution was adopted to Seshat, PoolParty, DBpedia, Jurion, and Jurion IPG. In the latter use case, it was used to ensure that the results of the validation processes carried out by Dacura and Semantic Booster be evaluated, manage this mapping and transformation, and save the transformed data to a triple store.

As regards the design phase in software engineering life cycle, the evidence for this benefit can be seen particularly strongly in the automated harvesting and curation interface generation tools developed in the project. This is particularly evident in case of Dacura that informs the software engineering analysis phase by defining what data is to be harvested (Shah et al. 2017: 381). The Dacura Linked Data Model Mapping Service tool creates rich ontological models from semi-structured HTML and automates harvesting of data conforming to this model and was heavily tested within the Jurion IPG use case.

For the implementation phase, the University of Leipzig developed a set of tools around RDFUnit and DataID, which together formed the basis for the Data Quality Framework and the Automated Data Testing and Verification Framework. Dacura makes it possible to define statistical data quality measures to be met to support software engineering and suggest UI refinements to eliminate errors. Repair Framework and Notification tool is used in both implementation and maintenance phase as the defined data constraints influence the implementation of algorithms and as taxonomies are changes, the constraints need to be satisfied.[14]

Semantic Booster tool allows the automatic generation of software systems from formally specified system specifications. Hence, it supports both semantic domain models and models of the software and data engineering life cycles. In particular, 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 gets 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. A Booster specification is

[14] Shah et al. 2017: 380.

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

This system was deployed in Jurion IPG and Health Data. The use of Semantic Booster in Jurion IPG allows the introduction of a wider range of semantic integrity constraints and business rules, to be applied on the data upon entry – ensuring availability of high-quality data. The automatic data migration tools provided with Semantic Booster minimise the impact of upgrading and evolving the underlying data model whilst still maintaining data consistency. Whilst Semantic Booster can already help enforce a range of integrity constraints, there are some consistency checks, which would be more reliably performed using RDF and reasoning. Hence, it was decided to use the existing D2RQ tool to convert data stored within a Booster database into RDF format, making it available to the RDFUnit testing tool. The additional testing and monitoring also provides insight into productivity and quality gains through the use of the ALIGNED tool stack.

The effective maintenance can be achieved in two alternative ways. The first approach is provided by a configuration of the Oxford MDE approach, while the second is by Dacura. A Booster specification is created, which (i) 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 and (ii) the Booster specification should ensure that this extracted data are correct by construction according to the Booster specification. Then, this extracted data are made available as RDF via Semantic Booster. In the approach offered by Dacura services, the Model mapping tool transforms the SQL schema of the legacy DB into an OWL ontology, which is then used by the schema checking tool to ensure that all data conforms to the model. The curation and workflow tools allow data managers to change the model and migrate the data and manage the process. This ontology is deployed as the schema for the graph into which the instance data are imported.

2.5 Conclusions

All five use cases in the ALIGNED project were thoroughly analysed to achieve its major goal, namely to create effective methods and tools for integrating software and data engineering processes and develop full life cycle workflows for combined software and data engineering. The deployment of the project designed and produced software and tools led to significant

enhancement of all case studies and significant improvements in data productivity, quality, and agility and eventually user satisfaction and customer support. In Jurion IPG, of particular significance turned out to be Semantic Booster, showing significant improvements in agility, with the addition of new attributes being up to 45 times faster. Also, Dacura significantly improved the management of re-engineering from the old relational database schema to the new one. In addition, Wolters Kluwer's Jurion and Jurion IPG business information database was enhanced with ALIGNED tools, significantly improving their ability to correct errors and change data schemas over their previous tools.

The introduction of import validation in the PoolParty use case improved data quality and reduced customer support time as well as significantly contributed to the ability to fix a number of violations. Overall improvements in data curation, data agility, model agility, and software development processes were also achieved. The major achievement in DBpedia was the error rate improved.

The rebuilding of the Seshat data and tools used the full suite of Dacura tools to import the data, ensure it met consistency requirements, automatically produce user interfaces and curation tools, and finally publish the data. It resulted in a quantifiable reduction in the number of errors in data entry, while the amount of data entered dramatically increased. The new format of the dataset enabled the ability to link to external datasets to enrich the Seshat data. The data generators and users reported an increase in usability and productivity, and the technical users reported an increase in agility: the speed in which tools and data can adapt to changes in the model. The shared model for data validation and software generation involves integration points with the planning phase of the software engineering life cycle, and the quality analysis, manual revision/authoring, and search/browse/explore phases of the data engineering life cycle. In addition, the collaborative consensus required for updating the model brings additional dependencies on the interlinking and extraction phases of data engineering.