Capturing Expert Knowledge for Building Enterprise SME Knowledge Graphs

Martin Mansfield
martin.mansfield@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Valentina Tamma
v.tamma@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Phil Goddard
phil.goddard@csols.com
CSols Ltd.
Runcorn, UK

Frans Coenen
coenen@liverpool.ac.uk
University of Liverpool
Liverpool, UK

ABSTRACT

Whilst Knowledge Graphs (KGs) are increasingly used in business scenarios, the construction of enterprise ontologies and the population of KGs from existing relational data remains a significant challenge. In this paper we report our experience in supporting CSols (an SME operating in the analytical laboratory domain) in transitioning their data from legacy databases to a bespoke KG. We modelled the KG using a streamlined approach based on state of the art ontology engineering methodologies, that addresses the challenges faced by SMEs when transitioning to new technologies: lack of resources to devote to the transition, paucity of comprehensive data governance policies, and resistance within the organisation to accepting new practices and knowledge. Our approach uses a combination of UML diagrams and a controlled language glossary to support stakeholders in reaching consensus during the knowledge capture phase, thus reducing the intervention of the ontology engineer only to cases where no agreement can be found. We present a case study illustrating the generation of the KG from a UML specification of part of the analytical domain and from legacy relational data, and we discuss the benefits and challenges of the approach.

CCS CONCEPTS

• **Computing methodologies → Ontology engineering**; *Model development and analysis.*

KEYWORDS

Ontology Engineering, Enterprise Knowledge Graphs, Relational Data, R2RML, UML

ACM Reference Format:
Martin Mansfield, Valentina Tamma, Phil Goddard, and Frans Coenen. 2021. Capturing Expert Knowledge for Building Enterprise SME Knowledge Graphs. In *Proceedings of the 11th Knowledge Capture Conference (K-CAP '21), December 2–3, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3460210.3493569

1 INTRODUCTION

The problem of designing enterprise ontologies from relational databases (RDBs) has received some attention in the literature, e.g. in [26]. Several ontology engineering methodologies have been proposed in the last 30 years, and have contributed to the area's methodological issues including the notion of *competency questions*, ontology reuse, and ontology design patterns [13, 16, 28, 29]. However, existing methodologies lack guidelines for building ontologies by reusing and re-engineering existing knowledge-aware, *non-ontological resources* [11], e.g. RDBs.

The reuse of RDBs typically involves domain experts (database administrators or SQL practitioners), often working with knowledge engineers (*knowledge scientists*[1]), to explicitly model knowledge embedded in RDBs. In most cases, this process requires the understanding and interpretation of business information and rules from relational schema, and the manual re-engineering of this knowledge in an ontological format [11, 18].

The problem of constructing ontologies from RDBs has also gained prominence in the context of Ontology Based Data Access (OBDA) approaches for implementing Knowledge Graphs, where *mappings* are defined between the *data sources* (i.e. RDBs) and the *ontology*, to provide a formal and abstract representation of the domain of discourse, that becomes the main interface of the information system [31]. The specification of mappings might be semi-automated, as in [3], where a rule based approach to the defining mappings based on table patterns aims to *learn* ontologies from RDBs. So far, approaches to the reuse of RDBs in the engineering of ontologies rarely consider the life cycle of building an ontology, or the associated software engineering requirements. Recently, some approaches, e.g. the "Pay-As-You-Go" (PAYG) approach [25], take a practice-based stance on the design and population of enterprise ontologies (*enterprise knowledge graphs*) from RDBs. The PAYG approach aims to address: (i) the inherent difficulty of engineering ontologies for a complex domains; and (ii) that it is not viable

[1] A recent essay (https://www.knowledgescientist.org) proposes the new organisational role of the knowledge scientist. This role encompasses the skills of a knowledge engineer, and liaises with data engineers and data scientists, to ensure reliable, clean, and meaningful data within an organisation.

to automatically populate an ontology from RDBs by casting the problem as a straightforward ontology mapping problem.

Capturing domain knowledge from enterprise databases is typically hindered by complex schema that encode assumptions about the domain in an opaque and ad-hoc way. Automatic approaches seldom manage to capture the complexity of a domain, and manual approaches require a “communication bridge” between business users and database specialists. The PAYG approach proposes the iterative development of an ontology in parallel to the definition of mappings between the ontology and relational data, and uses informal *intermediate representations* to close the gap between business requirements and data. We leverage on this approach in order to build a KG using a streamlined development process, which aims to address some of the challenges faced by small and medium enterprises (SMEs) in transitioning to semantic technologies for the organisation and exploitation of their data, mainly due to lack of resources and paucity of data governance policies.

This paper reports on the experience of building an ontology for an analytical software company, CSols. The ontology is constructed incrementally, using a streamlined engineering approach based on the well-established METHONTOLOGY [9], and the more recent PAYG methodology [25]. In Section 2, we identify our motivation and requirements for the design of a suitable engineering approach, and outline challenges specific to SMEs in engineering an enterprise ontology. We outline our proposed approach in Section 3, and concretely illustrate the approach with a case study in Section 4.

2 MOTIVATION AND CHALLENGES

This paper is born out of the experience acquired in the last 18 months working with CSols Ltd to engineer an enterprise KG to represent data and knowledge about analytical laboratories, their resources, and their processes. CSols is an SME that provides software solutions for the integration, analysis, and reporting of analytical data. In this section, we consider the motivations for the development of a novel approach to ontology engineering, and the specific challenges faced in the development of a CSols KG.

2.1 Motivation and Requirements

As part of their software portfolio, CSols maintain a collection of bespoke database (DB) systems that feed into their software solutions for data integrity and quality control, data reporting and integration. CSols took the strategic decision to build a KG to model and enrich the data currently stored in their DBs, to answer more sophisticated business questions and perform complex analytical tasks concerning instruments, their status, and their data output. Due to the nature of CSols business sector, their business questions typically involve specialised knowledge about chemistry, pharmacology and biotechnology, often widening the communication gap between stakeholders, as became evident when gathering requirements.

Knowledge capture is a critical stage of any methodology for building ontologies, and includes *specification* and *conceptualisation* activities, where requirements and usage scenarios are determined by the main actors of the methodology [2].

[2]Here we refer to the nomenclature used in METHONTOLOGY [9], but similar phases can be found in [20, 28, 29], just to name a few.

Knowledge about the domain in which CSols operates is embedded in legacy DBs and software. The relational schemas underpinning CSols’ products have evolved over decades, and are large and complex; they have been developed according to a range of programming styles and standards, and include arbitrary naming conventions and enumerations, as well as complex relations. The main interaction with these DBs is via a set of historical SQL queries that represent core business questions for the company, which are typically executed by *business users*. Business users are knowledgeable about aspects of analytical chemistry and are proficient in the use of SQL for querying DBs [27]. The design of more complex queries is supported by *IT developers*, who design, build and maintain CSols software and DB schemas. A significant challenge is to mediate between business users and IT developers and support effective communication between them in the contribution of requirements and constraints to the knowledge acquisition phase.

The analytical domain is well suited to being modelled as a KG; the widespread use of ontologies in the biochemical domain provides a wealth of existing ontological resources that can be used to model parts of the domain, such as the Chemical Methods Ontology (CHMO)[3], and Allotrope[4], a collection of ontologies developed by an international consortium of scientific industries for the management of laboratory data throughout its life cycle. These resources model different facets of the domain and have different levels of standardisation. However, there remains large areas of the domain for which there is no standardisation.

Following consultation with both business users and IT developers, we identified a number of core requirements to inform the choice of a suitable ontology development approach from a selection of the most prominent methodologies and processes [9, 20, 25, 28]. An appropriate approach must (i) facilitate knowledge capture from business users and software developers alike, by promoting a common understanding of the domain and its associated requirements; (ii) provide an explicit mechanism for eliciting stakeholder consensus; (iii) support the incremental development of the ontology: due to size and complexity of the domain, it was imperative that the ontology was developed incrementally, therefore rationalising the commitment of resources to the ontology development process; (iv) establish high-level interoperability with existing ontological resources describing facets of the analytical chemistry domain; (v) support at least RDFS (without sub-properties) expressivity, possibly extended by basic OWL primitives (`owl:Class`, `owl:DatatypeProperty`, `owl:ObjectProperty`, subsumption and domain and range declaration).

2.2 Challenges

Ontology engineering is inherently difficult, although there are many well-established methodologies aimed at tackling this complexity, such as METHONTOLOGY [9]. Here, we consider additional challenges posed by the design and instantiation of an ontology from legacy data, and those challenges specific to SMEs undertaking such activities.

2.2.1 *Engineering knowledge graphs from legacy data.*

The majority of ontology engineering methodologies consider the development

[3]https://github.com/rsc-ontologies/rsc-cmo

[4]https://www.allotrope.org/ontologies

of an ontology and its population as separate activities. In the case of developing a KG from RDBs, both the ontology and corresponding mappings from relational schema must be developed holistically. When developing mappings between relational schema and an ontology, the scale and complexity of enterprise DBs means that mappings between tables/classes and columns/properties are rarely one-to-one, and often include calculations to implement some business logic. For this reason, automatic approaches to mapping specifications (such as bootstrapping a putative ontology from DB schema and treating mappings as a mere ontology-matching problem) are generally impracticable. The iterative approach proposed in the PAYG methodology aims to overcome these challenges by focusing on a business question at time, and simultaneously developing both the ontology and mappings.

2.2.2 *Paucity of comprehensive data governance policies.*

Data governance refers to those data management techniques that enable an organization to ensure high data quality throughout the complete life cycle of the data, and that business objectives are supported through the implementation of appropriate data controls. However, data governance solutions are largely directed at the needs of large enterprises, and it has been argued that SMEs—with their limited resources—often have different requirements. A common assumption in industry is that data governance solutions for large enterprises can be scaled down and applied to SMEs, whereas SMEs need adaptable solutions that need to take into account the scarcity of available resources [2]. This typically translates into data resources (e.g. DB schemas) that are unnecessarily complex and often incomplete, and these legacy issues are often difficult to eradicate. Any methodological solution for the development of ontologies in this scenarios needs to include mechanisms to overcome these limitations. In this type of scenario, developing tools to semi-automate part of the development process can speed up the process and limit the role of the knowledge scientist to focus on those issues that require their specific expertise.

2.2.3 *Acceptance of new technologies and knowledge.*

Managing change in SMEs is typically done in-house, by the business owner, with no specialists, no dedicated departments, and no budget for external consultancy. However, the business owner often has to overcome a reluctance to change from employees. It has been argued that a key factor in successfully implementing change is the use of effective communication mechanisms as a way to improve participation and ownership, and promote taking common action [17]. We argue that approaches to building ontologies need to ensure that clear and unambiguous communication mechanisms are in place throughout the specification and conceptualisation phase, together with mechanisms to elicit consensus between stakeholders with divergent or conflicting views.

3 PROPOSED APPROACH

The development process we propose is loosely based on the PAYG methodology proposed by Sequeda and colleagues [25]: it is similarly iterative and facilitates the simultaneous development of both an ontology and RDB2RDF mappings. Each iteration is based on determining a set of prioritised concepts through a set of crucial competency questions formulated by the key actors involved the development process. In CSols, we identify two main actors: *business users* who in this case are domain experts, with knowledge of analytical laboratories, their resources and their processes, and *IT developers*, who understand DB schemas and data organisation.

Given the relatively small size of CSols, the transition to semantic technologies has to be done in-house, with minimal reliance on external resources. To minimise the use of specialist resources, we delegate the responsibilities of a knowledge scientist to two alternative roles. Firstly, an *ontology engineer*, with expertise in ontological modelling and a responsibility for knowledge transfer activities; this role also acts as a mediator in consolidating conflicting ideas when capturing domain knowledge from business users. Secondly, a *data manager*, with expertise in the type of data used throughout the company and how it is integrated, and an understanding of the evolution of relational schema over time. In our case, the data manager is a CSols employee who was able to undertake many of the ontology engineering tasks with their existing skills and resources. This limits the intervention of the ontology engineer to the most complex cases.

The proposed ontology development process is articulated in three phases: knowledge capture, knowledge implementation, and knowledge exploitation.

3.1 Phase 1: Knowledge Capture

Each iteration of the CSols ontology development approach augments the ontology to include the knowledge required to address specific business concerns, typically identified through the formulation of competency questions. The approach is incremental and modular, where each module aims to capture the knowledge abstracting the data held in one or more of the CSols database tables. The Knowledge Capture Phase aims to identify *business questions* and the data necessary to answer them through the analysis of existing business processes and the collection of any supporting documentation, including the identification of ontologies to reuse.

During this phase, stakeholders collectively define the concepts related to a particular business process, their relationships and constraints. UML Class Diagrams are used as used as a lingua franca between chemists, laboratory workers and software developers, iterating over definitions until consensus is reached.

Process Analysis The first step is to analyse and formalise a particular process undertaken by CSols and understand the specific business concerns addressed by it. The different perspectives of both business users (in our case laboratory workers, chemists and commercial sales representatives) and IT developers are considered in order to establish i) the problem to be solved by the specific process; ii) the business rational behind the process; iii) the way the problem is addressed by legacy data, if at all; and iv) any other processes producing and consuming the relevant data.

Once the business concern is understood, the data manager can collate any existing documentation describing how it is currently addressed. For CSols, this typically includes product documentation and SQL queries either commonly executed by business users or embedded in software.

Conceptualisation with UML. The next step is to conceptualise facets of the domain related to the identified business process. This includes the identification and precise definition of key concepts and the relationships between them. The definition of domain concepts and relationships provides an *intermediate representation* of the domain, and gives stakeholders an informal view of the domain that can be understood by both subject matter experts and the ontology engineer, bridging the gap between the business users' understanding of the domain and the formal ontology language used to represent it.

In this step, the data manager analyses any collected documentation to draft an intermediate representation, which is shared with business users and IT developers for further refinement. Disagreements are common during this process, with stakeholders using the same word to describe different concepts or different words to describe the same concept [25]. Many of these conflicting terms are embedded in software and data schema, and contribute to the challenge of querying complex enterprise data. Given the iterative development of the ontology, additional conflicts might be introduced between new and established definitions. Where stakeholders are unable to reach consensus, focused intervention by the ontology engineer can help to disambiguate between the different meanings using formal models.

Similarly to the intermediate representations introduced in the conceptualisation phase of METHONTOLOGY [9], the PAYG approach proposes a tabular *knowledge report* to record agreed definitions, with a collection of tables recording information about each concept, their attributes, and relationships between them[5]. When applying the PAYG approach to CSols, a tabular representation was ineffective for precisely capturing the complexity of the domain and communicating information about identified concepts between diverse stakeholders. We therefore propose the use of UML Class Diagrams to support the specification of intermediate representations. A de-facto standard formalism for software design and analysis, UML is widely used to model application domains and communicate effectively, and is sufficiently abstract to be understood by business users, IT developers, data managers, and ontology engineers.

The use of UML as an ontology modelling language is long-established [6]. Parallels between UML and formal ontologies have motivated research investigating their combination in integrated approaches, including the validation of UML models using automated ontology reasoning techniques (e.g. [4, 15, 24]), the organisation and unification of complementary domain models (e.g. [14, 23]), and the use of UML in the design and specification of ontologies (e.g. [5, 12, 21, 33]). The Class Diagram is most commonly used in the elicitation of domain knowledge, and includes constructs closely aligned with those included in formal ontology languages. The class diagram facilitates the specification of *classes* which can include *attributes.* Additional constructs enable the specification of relationships between classes, including *generalisation*, *association*, and *aggregation.* Using these constructs, class diagrams provide mechanisms to encapsulate the Concepts, Attributes, and Relationships embodied a the tabular Knowledge Report, and thus can be used instead of a Knowledge Report. In our experience, the use of UML during the conceptualisation phase can afford a range of benefits, including (i) the semi-formal syntax of UML aids stakeholders in capturing the complexity of the domain; (ii) visualisation of concepts and relationships supports stakeholders in understanding the domain and significantly reduces the effort and intervention required in reconciling conflicting perspectives; (iii) UML views enable stakeholders to aggregate concerns related to a particular use case or business question; (iv) widely available CASE tools aid the organisation of complementary views and automate consistency checking between them.

Reuse of Existing Ontologies. In this step, the ontology engineer works with business users to align the ontology with existing ontologies relevant to the analytical laboratory domain. Concepts included in external resources, or related to concepts in external resources are identified and reused. Following [10], we identify two modalities of reuse: *hard reuse*, where an external resource is imported to the ontology, and *soft reuse*, where the URI of an external element is referenced in the ontology. The appropriateness of the type of reuse is assessed for each case, with an aim of maintaining focus on the domain of interest.

3.2 Phase 2: Knowledge Implementation

In the Knowledge Implementation Phase, intermediate representations of the domain are formalised in an OWL ontology and corresponding mappings from relational schema to RDF are defined using R2RML[6]. We propose an automated transformation from UML to OWL/R2RML, ensuring a systematic translation and further reducing the intervention of an ontology engineer. Following transformation, the ontology and mappings are evaluated by the definition and execution of appropriate SPARQL queries.

Model Transformations. In this step, intermediate representations are formalised using a formal ontology language. The similarities between UML and ontology languages enable a direct mapping between models.

The integration UML and ontology languages, chiefly between UML Class Diagrams and OWL [19], is typically obtained through syntactic transformations. Though inherent differences in these languages prevent a complete mapping between them, many components of the Class Diagram can be mapped directly to OWL constructs. The provision of a UML profile can enable the extension of the Class Diagram to include additional stereotypes to support modelling additional OWL constructs (e.g. [1, 7]), however these extensions result in a UML variant more specific to ontology modelling, with additional learning requirements for users and overheads for CASE tool integration.

El Hajjamy et al. [8] summarise 8 different methods for mapping UML to OWL, and use the combination of these techniques as a basis for the specification of a comprehensive set of transformation rules. Additional rules are defined by Vo et al. [30] for mapping additional constructs, including structured attributes, recursive association, association with association classes and qualified aggregation. Based on these rules, we propose a transformation framework summarised in Table 1, which outlines a mapping between UML and OWL.

We further enrich intermediate representations to support the generation of R2RML mappings. We propose the use of UML Constraints to embed SQL queries directly in the UML model, with constraints added to both Classes and Attributes. The correspondence between UML and R2RML constructs is summarised in Table 2.

[5]Here, *Concepts*, *Attributes* and *Relationships* correspond to Classes, Datatype properties and Object properties typical of OWL ontologies.

[6]https://www.w3.org/TR/r2rml/

Table 1: Correspondence between UML and OWL constructs

UML Construct	OWL Ontology
Class	owl:Class
Class Name	rdfs:label of the Class
Comment	rdfs:comment of the Class
Concept Name	Used to create the URI of the Class
Attribute	owl:DatatypeProperty
Attribute Name	rdfs:label of the Datatype Property
Comment	rdfs:comment of the Datatype Property
{concept-attribute}	Used to create the URI of the Datatype Property
Host Concept	rdfs:domain of the Datatype Property
Type	rdfs:range of the Datatype Property
Generalisation	rdfs:subClassOf
Association	owl:ObjectProperty
Association Name	rdfs:label of the Object Property
Aggregation	owl:ObjectProperty (with rdfs:label 'uses')
Composition	owl:ObjectProperty (with rdfs:label 'contains')
Comment	rdfs:comment of the Object Property
{domain-relationship-range}	Used to create the URI of the Object Property
From Class	rdfs:domain of the Object Property
To Class	rdfs:range of the Object Property

Table 2: Correspondence between UML and R2RML

UML Construct	R2RML Mapping
Classes	
Class Name	rr:Class
Class Constraint	used to generate rr:template
Class Constraint body	rr:logicalTable
Class Attributes	
Class Constraint (owning class)	used to generate rr:template
Attribute Constraint	used to generate rr:column
{concept-attribute}	rr:predicate
Attribute Constraint body	rr:logicalTable
Associations	
Class Constraints (both From and To)	combined to generate rr:logicalTable
domain-relationship-range	rr:predicate
Class Constraint (From Class)	used to generate rr:template
Class Constraint (To Class)	used to generate rr:joinCondition

UML views are based on a metamodel which can be exposed using the XML Metadata Interchange (XMI)[7] standard for exchanging metadata information via XML. By exposing the metamodel using XMI, the transformation between intermediate representations and OWL/R2RML can be automated. Such transformation algorithms are demonstrated in [8]. This automation not only ensures a systematic translation between models, but further reduces the involvement of the ontology engineer.

Query Extraction and Ontology Verification. In this step, the ontology engineer inspects the formal ontology produced by model transformations, to ensure it adequately reflects the intermediate representation and can be used as an interface to legacy data.

The ontology engineer ensures that the resulting ontology is syntactically correct, and does not manifest those common problems that can occur when an ontology is developed collaboratively. They define a set of SPARQL queries to capture the information of interest to business users, based on existing SQL queries used to address the business concern. The execution of these SPARQL queries should return the same data as one or more SQL queries.

The combination of the OWL ontology and R2RML mappings facilitates the execution of SPARQL queries against existing relational data using ODBA systems, that provide access to data stored in relational DBs via a virtualised knowledge graph; they aggregate the data according to a given ontology and translate SPARQL queries to SQL using RDB2RDF mappings [31]. The results returned by SPARQL queries are validated by comparison against the results of equivalent SQL queries.

3.3 Phase 3: Knowledge Exploitation

A KG is typically populated by mapping legacy data to a domain ontology. In the Knowledge Exploitation Phase, business users analyse the results of querying the knowledge graph to confirm that it can be used to satisfactorily address the business questions defined in Phase 1. This confirmation enables the ontology to be released for use by the wider company, and the ontology can be developed in a further iteration. We describe how CSols exploit the resulting knowledge graph by enabling access to legacy data via a SPARQL endpoint, and by using the controlled vocabulary to aid communication of complex domain concepts between diverse stakeholders.

Knowledge Graph Validation. In this step, the ontology engineer works with business users and IT developers to validate the knowledge graph and SPARQL queries. The ontology and/or queries are refined until all stakeholders are satisfied that query results both align with the results of existing SQL implementations, and adequately address the business concern identified in Phase 1. Once consensus is reached, the ontology iteration can be moved to a production environment.

Move to Production. Once stakeholders agree that the ontology sufficiently addresses the identified business concerns, it can be released into a production environment. In [25], the focus of this step was integration with business intelligence tools. In the CSols case, we focus on the provision of a SPARQL endpoint for both manual queries by business users, and for IT developers to exploit in software. By enabling the use of SPARQL to query legacy data, queries are expressed in terms of the domain which are formally defined, and require no knowledge about the schema used to store the data. Querying a graph model enables data exploration, data federation, and schema reuse.

4 CASE STUDY

In this section we present a case study which demonstrates the approach proposed in Section 3 applied to the CSols scenario of developing an ontology describing facets of analytical chemistry.

CSols products are underpinned by a common relational database, which integrates data directly from analytical instruments, manual data entry, and from software. The majority of this data describes results from the analysis of samples, and the resources used in the generation, transcription, and storage of the those results.

The business concern addressed in this study is related to the identification of laboratory resources. Laboratory resources include hardware, software, and people, and data about each resource varies significantly. In this study, we model a hierarchical taxonomy of

[7] https://www.omg.org/spec/XMI/2.5.1/PDF

resources and their role in the analytical process. Legacy relational data is mapped to the ontology to enable the execution of SPARQL queries against a virtualised KG.

We evaluate the ontology by verifying the query results against existing SQL queries, and ensuring that the results adequately address the business concern. We evaluate the approach by consideration of its merit compared with directly querying relational data.

4.1 Knowledge Capture

In the Knowledge Capture Phase, we establish and clarify the business concern to be addressed and the identify data needed to answer relevant questions. We analyse existing processes and mechanisms to obtain this data and model key concepts and relationships using UML. once these concepts are defined, we align our definitions with those in related ontologies.

Process Analysis and Documentation Collection. In this phase, a single critical business concern is identified by business users; they agree to focus on a process to identify all types of a resources deployed in a given laboratory. This is an important query often used in a variety of auditing activities and is currently implemented by a series of SQL scripts which separately target a collection of tables known to each store information about a type of resource (e.g. people, instruments, software). Whilst the notion of a generic 'resource' is commonplace throughout auditing activities, it is not modelled in the existing schemas. Retrieving this information depends on exhaustively analysing the schemas after every update, to isolate any tables related to resources, and collating the results of several complex queries to aggregate the data into a single report.

With the business concern established, the data manager collaborated with business users and IT developers to collate documentation about existing solutions, including DB schemas, queries, and scripts. The ER diagram in Figure 1 illustrates a subset of the relational schema underpinning existing solutions.

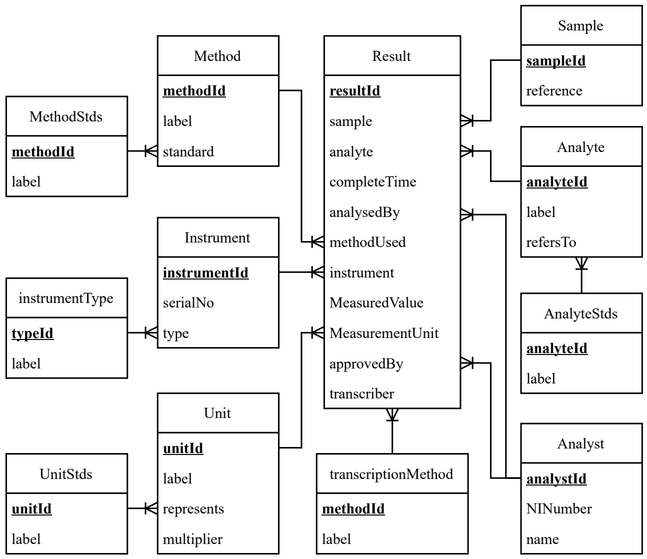

Figure 1: Entity Relationship Diagram of legacy data schema

Conceptualisation with UML. With documentation about existing processes collated, a preliminary set of UML Class Diagrams were drafted by the data manager. This intermediate representation defines key concepts, their attributes, and relationships between them, and was iteratively refined by both business users and IT developers until all stakeholders agreed on each definition. The Modelio[8] open source tool was used to build and share model since it supports UML with model assistance and automated consistency checking. Figure 2 shows an example Class Diagram illustrating the hierarchical categorisation of laboratory resources.

Reuse of Existing Ontologies. An analysis of the intermediate representation identified several related ontologies suitable for reuse. We reuse the Quantities, Units, Dimensions and Types (QUDT) ontology[9] for describing units of measurement, the Chemical Methods Ontology (CHMO)[10] for describing analytical techniques, the Chemical Entities of Biological Interest (CHEBI) ontology[11] for describing chemical composition, and Allotrope[12]. Of these ontologies, we directly import QUDT, and create explicit mappings to specific concepts in CHMO, CHEBI and Allotrope. The CSols ontology is by nature more specific than the ontologies reused indirectly, which are used to ensure interoperability with other commercial products.

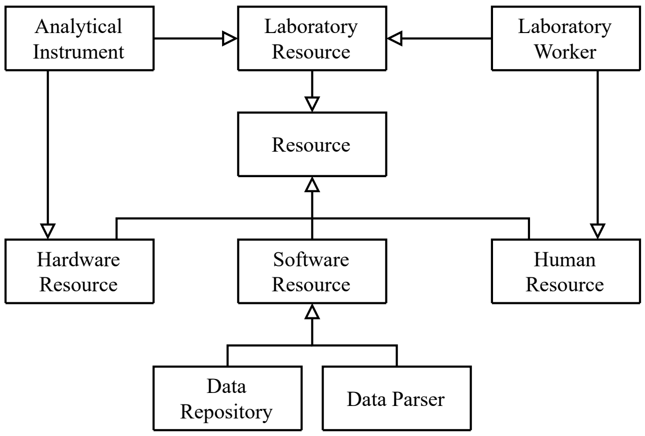

Figure 2: Class Diagram modelling laboratory resources

4.2 Knowledge Implementation

In the Knowledge Implementation Phase we transform the UML intermediate representation into a formal ontology and corresponding R2RML mappings. SPARQL queries are defined and used to verify that the resulting virtual knowledge graph is consistent with directly querying relational data.

Model Transformations. We automate the transformation rules outlined in Section 3 by exporting the XMI metamodel and parsing model elements using the Python *ElementTree* API[13]. The parser outputs an OWL ontology, with definitions such as:

[8]https://www.modelio.org/
[9]https://www.qudt.org
[10]https://github.com/rsc-ontologies/rsc-cmo
[11]https://www.ebi.ac.uk/chebi/
[12]https://www.allotrope.org/ontologies
[13]https://docs.python.org/3/library/xml.etree.elementtree.html

```
:sample rdf:type owl:Class ;
    rdfs:comment "A sample of a material to be analysed" ;
    rdfs:label "Sample" .
:sample-reference rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:domain :sample ;
    rdfs:range xsd:string ;
    rdfs:label "reference" . }
```

The Python parser also implements transformation rules for generating R2RML mappings, such as:

```
map:m1 a rr:TriplesMap;
  rr:logicalTable [ rr:sqlQuery "SELECT sampleId from Sample" ];
  rr:subjectMap [ rr:class :sample; rr:template
      "http://ontologies.csols.com/data/Sample/{sampleId}" ] .
```

For the given iteration, the UML model is translated to an ontology with 24 classes, 13 object properties, and 11 data properties.

Query Extraction and Ontology Validation. The resulting ontology was assessed using OOPS! (OntOlogy Pitfall Scanner) [22] to ensure that it did not contain errors. Once satisfied with the quality of the ontology, a series of SPARQL queries were defined and executed by the ontology engineer, based on those SQL queries identified in Phase 1. By comparing the results from executing SPARQL queries against the virtual KG with those from executing SQL queries against relational data, the the fidelity of both the ontology and mappings was verified. We use the open source Ontop OBDA tool [32] to execute the SPARQL queries against relational the data stored in legacy SQL Server RDBs. The resulting virtual KG is populated with 281 individuals when the ontology is mapped to a small legacy dataset specifically selected for validation.

4.3 Phase 3: Knowledge Exploitation

In the Knowledge Exploitation Phase, we finalise queries which retrieve data related to the business concern identified in Phase 1, and move the ontology into production once business users are satisfied adequately addresses the targeted business concerns.

Knowledge Graph Validation. This iteration of the CSols ontology identifies all resources utilised in the generation of a set of results. In the schema described, data is collected that describes a range of resources, including analytical instruments, software, and human analysts. Information about each resource type is distributed across different tables, and there is no overall notion of a generic resource. Attaining a complete picture of all resources requires a detailed knowledge of the relational schema (or an exhaustive evaluation of it), and the aggregation of data from different tables for each resource type. By mapping existing data to the hierarchical resource taxonomy modelled in the ontology, this query is vastly simplified using SPARQL:

```
SELECT DISTINCT ?r {
  ?r rdf:type+ :resource .
}
```

By evaluating the results of SPARQL queries, business users confirmed that the virtual knowledge graph adequately addresses those business concerns identified in Phase 1.

Move to Production To finish the iteration, the ontology is made available across the company. Ontop is used to provide an endpoint for the execution of SPARQL queries against legacy data. This is used by business users in executing manual queries, and by IT developers, who embed queries in software. Existing SQL queries are replaced with SPARQL equivalents where performance improvements can be gained, or where code clarity is improved.

The ontology is also released to the company as a controlled vocabulary. This is used by stakeholders to maximise expressivity and minimise ambiguity when communicating complex domain concepts. From writing customer proposals to architecting new software solutions, the ontology is used to ensure uniformity in representing the domain.

5 DISCUSSION

The approach we presented in Section 3 relies on the ability to capture the constraints and requirements representing the world-view of the relevant stakeholder in a way that does not add to their cognitive load. Whilst the use case presented here draws on constraints that are domain specific, the approach we propose can be easily generalised to any domain. CSols decided to implement bespoke tools to translate from UML to OWL and to define the mappings from relational schemas through R2RML only based on a view to gain a commercial advantage by building a complete toolkit. However, these translations are based on existing literature and can be easily replicated, or existing open source tools can be used as an alternative, making the development process semi-automatic, thus introducing stricter data governance policies.

By using a familiar modelling paradigm, we mitigate against the reluctance of company employees to accept new technologies and knowledge. This was the motivation for using only UML class diagrams, with no profile extensions. The drawback of such a simplistic model is that some ontological nuances are missed in the knowledge capture phase, e.g. non disjoint classes, that would have been better supported by an ontology-based conceptual modelling tool such as OntoUML [12]. However, this allowed us to tightly scope the discussions between business users and IT developers during this phase. Another important factor that supported this decision is the choice of expressivity of the final ontology, which in this case is mainly RDFS extended with basic OWL primitives. This of course caused some modelling errors that were rectified by the intervention of the ontology engineer. For instance, when modelling laboratory resources it was realised that both a data parser and a laboratory worker can move data between resources, and therefore be types of data transporters. An initial version of the model included the data parser and the laboratory worker as non-disjoint subclasses of the class Data-Transporter-Role, but in a subsequent iteration, and after the intervention of the ontology engineer, a "hasRole" object property was introduced [14].

The proposed process allowed us to constrain the time devoted to knowledge capture, thus addressing one of the concerns faced by SMEs: having only limited resources to buy in new technology and expertise. In the presented use case, each of the 24 classes and associated properties required an effort equivalent to 0.54 person months from the business user and IT developer, and each class underwent on average 2 rounds of discussions. The ontology engineer's intervention was limited to supporting more foundational

[14]The CSols ontology contains commercially sensitive material and therefore cannot be shared

decisions, such as avoiding the misuse of the subclass relation or defining disjointness axioms, and can be quantified in 5 hours.

We believe that the definition of a glossary together with the conceptual model helps resolve conflicts arising when definitions in the legacy DBs clashed with those of the newly defined data, and to document these decisions. The glossary is a resource that aids communication of complex domain concepts between business stakeholders, supports the design of software architecture, and is meant to be provided to CSols customer as a further resource.

6 CONCLUSIONS

In this paper we presented our experience in building an enterprise KG for the area of analytical laboratories as an in-house effort by an SME and we identified the particular challenges that SMEs face in producing such KG, specifically (i) paucity of comprehensive data governance policies; and (ii) acceptance of new technologies and knowledge. These challenges directly result from the limited resources that SMEs can devote to innovation, a well-documented problem in organisational management literature [2, 17].

In response to the identified challenges, we modified the PAYG approach [25] to limit the intervention of the ontology engineer to those complex cases where no agreement could be reached amongst the stakeholders. For the simpler cases, domain knowledge is acquired through the specification of an intermediate graphical model using UML Class Diagrams.

We evaluated the proposed approach by means of a case study that demonstrates how stakeholders with limited ontological knowledge were able to collaboratively construct the ontology and corresponding R2RML mappings with minimal intervention by the ontology engineer. The ontology was validated through SPARQL queries matching the competency questions identified in the knowledge capture phase of the approach. We argue that the use of a graphical intermediate representation that facilitates the semi-automatic development of the OWL representation is a viable way to lower the barrier preventing SMEs from stepping onto the semantic technology ladder, especially in those cases where legacy data is involved.

REFERENCES

[1] Kenneth Baclawski, Mieczyslaw K Kokar, Paul A Kogut, Lewis Hart, Jeffrey Smith, Jerzy Letkowski, and Pat Emery. 2002. Extending the Unified Modeling Language for ontology development. *Software and Systems Modeling* 1, 2 (2002), 142–156.

[2] Carolyn Begg and Tom Caira. 2012. Exploring the SME quandary: Data governance in practise in the small to medium-sized enterprise sector. *Electronic Journal of Information Systems Evaluation* 15, 1 (2012).

[3] Bilal Ben Mahria, Ilham Chaker, and Azeddine Zahi. 2021. A novel approach for learning ontology from relational database: from the construction to the evaluation. *Journal of Big Data* 8, 1 (2021), 25.

[4] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. 2005. Reasoning on UML class diagrams. *Artificial intelligence* 168, 1-2 (2005), 70–118.

[5] Sara Brockmans, Raphael Volz, Andreas Eberhart, and Peter Löffler. 2004. Visual Modeling of OWL DL Ontologies Using UML. In *The Semantic Web – ISWC 2004*. Springer, Berlin, Heidelberg, 198–213.

[6] Stephen Cranefield and Martin K. Purvis. 1999. UML as an Ontology Modelling Language. In *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration (CEUR Workshop Proceedings, Vol. 23)*. Stockholm, Sweden, 1–8.

[7] Dragan Djurić, Dragan Gašević, Vladan Devedžić, and Violeta Damjanović. 2005. A UML profile for OWL ontologies. In *Model Driven Architecture*. Springer, 204–219.

[8] Oussama El Hajjamy, Khadija Alaoui, Larbi Alaoui, and Mohamed Bahaj. 2016. Mapping UML to OWL2 ontology. *Journal of Theoretical and Applied Information Technology* 90, 1 (2016), 126–143.

[9] Mariano Fernandez, Asuncion Gomez-Perez, and Natalia Juristo. 1997. Methontology: from ontological art towards ontological engineering. In *Proc. of the AAAI97 Spring Symposium Series on Ontological Engineering*. Stanford, USA, 33–40.

[10] Mariano Fernández-López, María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asuncion Gomez-Perez. 2018. Why are ontologies not reused across the same domain? *Journal of Web Semantics* 57 (2018), 1–9.

[11] Andrés García-Silva, Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa, and Boris Villazón-Terrazas. 2008. A Pattern Based Approach for Re-engineering Non-Ontological Resources into Ontologies. In *The Semantic Web*. Springer, 167–181.

[12] John Guerson, Tiago Prince Sales, Giancarlo Guizzardi, and João Paulo A. Almeida. 2015. OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies. In *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*. IEEE, 144–147.

[13] P. Hitzler, A. Gangemi, K. Janowicz, A. A. Krisnadhi, and V. Presutti. 2016. *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*. IOS Press, NLD.

[14] C. Maria Keet and Pablo Rubén Fillottrani. 2015. An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. *Data & Knowledge Engineering* 98 (2015), 30–53.

[15] Ali Hanzala Khan and Ivan Porres. 2015. Consistency of UML class, object and statechart diagrams using ontology reasoners. *Journal of Visual Languages & Computing* 26 (2015), 42–65.

[16] M.F. Lopez, A. Gomez-Perez, J.P. Sierra, and A.P. Sierra. 1999. Building a chemical ontology using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems* 14, 1 (Jan-Feb 1999), 37–46.

[17] Paula Matos Marques Simões and Mark Esposito. 2014. Improving change management: How communication nature influences resistance to change. *Journal of Management Development* 33 (04 2014), 324–341.

[18] Robert Meersman. 2001. Ontologies and Databases: More than a Fleeting Resemblance. *Springer, Rome Workshop, Luiss Publications*.

[19] Meriem Mejhed Mkhinini, Ouassila Labbani-Narsis, and Christophe Nicolle. 2020. Combining UML and ontology: An exploratory survey. *Computer Science Review* 35 (2020), 100223. https://www.sciencedirect.com/science/article/pii/S1574013719300231

[20] N. Noy and D.L. McGuinness. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical Report SMI-2001-0880. Stanford Medical Informatics (SMI), Department of Medicine, Stanford University.

[21] Fernando Silva Parreiras and Steffen Staab. 2010. Using ontologies with UML class-based modeling: The TwoUse approach. *Data & Knowledge Engineering* 69, 11 (2010), 1194–1207.

[22] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. 2014. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)* 10, 2 (2014), 7–34.

[23] Karina Robles, Anabel Fraga, Jorge Morato, and Juan Llorens. 2012. Towards an ontology-based retrieval of UML Class Diagrams. *Information and Software Technology* 54, 1 (2012), 72–86.

[24] Małgorzata Sadowska and Zbigniew Huzar. 2017. Semantic validation of UML class diagrams with the use of domain ontologies expressed in OWL 2. In *Software Engineering: Challenges and Solutions*. Springer, Wroclaw, Poland, 47–59.

[25] Juan F. Sequeda, Willard J. Briggs, Daniel P. Miranker, and Wayne P. Heideman. 2019. A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases. In *Proceedings of the 18 International Semantic Web Conference (ISWC 2019)*. Springer International Publishing, 526–545.

[26] Juan F. Sequeda, Syed Hamid Tirmizi, Óscar Corcho, and Daniel P. Miranker. 2011. Survey of directly mapping SQL databases to the Semantic Web. *Knowledge Engineering Review* 26, 4 (2011), 445–486.

[27] Nigel Shadbolt and Paul R Smart. 2015. Knowledge Elicitation: Methods, Tools and Techniques. In *Evaluation of Human Work*. CRC Press, 163–200.

[28] Mari Carmen Suarez-Figueroa, Asuncion Gomez-Perez, Enrico Motta, and Aldo Gangemi. 2012. *Ontology Engineering in a Networked World*. Springer.

[29] Mike Uschold and Michael Gruninger. 1996. Ontologies: Principles, methods and applications. *The knowledge engineering review* 11, 02 (1996), 93–136.

[30] Minh Hoang Lien Vo and Quang Hoang. 2020. Transformation of UML class diagram into OWL Ontology. *Journal of Information and Telecommunication* 4, 1 (2020), 1–16.

[31] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-based data access: A survey. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI-18*. Stockholm, Sweden, 5511–5519.

[32] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalaycı, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. 2020. The virtual knowledge graph system ontop. In *International Semantic Web Conference*. Springer, Cham, 259–277.

[33] Zhuoming Xu, Yuyan Ni, Wenjie He, Lili Lin, and Qin Yan. 2012. Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach. *World Wide Web* 15, 5 (2012), 517–545.