A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases

Juan F. Sequeda, Willard J. Briggs, Daniel P. Miranker, and Wayne P. Heideman

Capsenta Inc., Austin, Texas, USA
** juan,will,miranker,wayne@capsenta.com

Abstract. Business users must answer business questions quickly to address Business Intelligence (BI) needs. The bottleneck is to understand the complex databases schemas. Only few people in the IT department truly understand them. A holy grail is to empower business users to ask and answer their own questions with minimal IT support. Semantic technologies, now dubbed as Knowledge Graphs, become useful here. Even though the research and industry community has provided evidence that semantic technologies works in the real world, our experience is that there continues to be a major challenge: the engineering of ontologies and mappings covering enterprise databases containing thousands of tables with tens of thousands of attributes. In this paper, we present a novel and unique pay-as-you-go methodology that addresses the aforementioned difficulties. We provide a case study with a large scale e-commerce company where Capsenta's Ultrawrap has been deployed in production for over 3 years.

1 Introduction

Business users must answer critical business questions, optimize business decisions and deliver the answers as accurately and quickly as possible. The frequent bottleneck to delivering answers is the lack of understanding by business users of their large and complex enterprise databases. The handful of IT experts that fully understand the abstruse and inscrutable database schemas are not always available. Business users interfacing with both the IT staff and the database systems can entail communication problems due to lack of agreed terminology. The ultimate goal is to empower business users to achieve Self Service Analytics (SSA), which Gartner defines as "*a form of business intelligence (BI) in which line-of-business professionals are enabled and encouraged to perform queries and generate reports on their own, with nominal IT support.*"[1] Gartner predicts that this year, 2019, the analytics generated by business users through SSA capabilities will exceed the analytics produced by professional data scientists[2].

Gartner crisply states that '*Self-service analytics is often characterized by [...] an underlying data model that has been simplified or scaled down for ease of understanding and straightforward data access.*" It should be evident to the reader (coming from

** At the time of submission, Capsenta was an independent company. It was acquired by data.world in mid 2019.

[1] https://www.gartner.com/it-glossary/self-service-analytics/

[2] https://www.gartner.com/newsroom/id/3848671

the semantic web community) that semantic web technology, dubbed now as Knowledge Graphs, deliver precisely this value. Informally, we define a **Knowledge Graph** *as the result of data integration based on graphs where concepts and relationships are first class citizens and the data comes from inter-linking heterogeneous sources of data*[3]. Furthermore, we consider Ontology-Based Database Access (OBDA) [10] as a mechanism to implement Knowledge Graphs[4]. OBDA involve the development of an ontology as a formal conceptual model of a domain which uses the lingua franca of the business users and declarative mappings between select contents of the enterprise's databases and the ontology. The declarative mappings allow comfortable communication between business users and IT developers. Finally, business questions can be defined in terms of the ontology's logical abstraction instead of the individual heterogeneous source databases' physical structures. OBDA systems can support physical or virtual access to the data. Capsenta deploys Ultrawrap [11, 14] as an OBDA system to deliver business-friendly data in the form of a Knowledge Graph over the inscrutable and heterogeneous databases. The resulting data is used by business users to achieve Self-Service Analytics.

The Semantic Web community continues to constantly provide evidence that semantic technology works in the real world (e.g. In-Use and Industry papers at ISWC, ESWC, Knowledge Graph Conferences[5]; renowned enterprises such as Google, Oracle, etc; startups like Capsenta and data.world; European Projects such as Optique [7], etc). Despite these successes, leading experts in semantic web, from both academia and industry, agree that the technology is still hard to deploy[6] because "systems are built such that they require PhD-type users to maintain"[7]. Given our experience of designing and building Knowledge Graphs with large scale customers in the e-commerce, oil and gas, and pharmaceutical domain since 2015, we agree! It is paramount that we devise ways to lower the barrier to entry in order to see the usage of semantic technology accelerate in industry.

In this paper, we argue that ontology engineering methodologies need to be extended to support Relational Database to RDF Graph (RDB2RDF) mapping engineering (Section 2). ***Our main contribution is an iterative pay-as-you-go methodology, that builds on existing ontology engineering methodologies and combines the engineering of Relational Database to RDF Graph (RDB2RDF) mappings. To the best of our knowledge, this is the first methodology that combines ontology and Relational Database to RDF Graph (RDB2RDF) mapping engineering*** (Section 3). It is important to note that this work can be applied in a straightforward manner to schema and mapping engineering for Property Graphs.

[3] It is important to acknowledge that the term "Knowledge Graph" was coined in a marketing blogpost by Google in 2012. Nevertheless, this term encompasses the work of the Semantic Web community from the early 2000s, which builds on the research combining Logic and Data that can be traced to the 1970s. See http://knowledgegraph.today for more details.

[4] Which nowadays is being called Virtual Knowledge Graphs.

[5] https://www.eventbrite.com/e/2019-knowledge-graph-conference-tickets-54867900367

[6] https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html

[7] http://www.juansequeda.com/blog/2019/03/22/2nd-u-s-semantic-technologies-symposium-2019-trip-report/, https://www.open-bio.org/2019/04/06/us2ts/

We define success if a business question is answered. The methodology is a means to success (i.e. answering business questions). Every time there is an iteration, it's because 1) an existing set of business questions have been answered successfully and 2) there are a new set of business questions to be answered. Therefore every iteration is an indication that more business questions are being answered, hence success has been achieved. Success is achieved without "*boiling the ocean*".

This paper provides a case study and a real world example with a large scale e-commerce company as evidence of how our pay-as-you-go methodology is successfully used to engineer an ontology and mappings which drives data for Self Service Analytics. This solution has been in production for the past 3 years (Section 4). We conclude with lessons learned and future work that we believe can inspire scientific research (Section 5).

2 Challenges

Per our real-world experience, the main challenges that must be overcome to design and build Knowledge Graphs from relational databases are **(C1)** Ontology Engineering: designing the ontology that models the domain experts's view of the world and **(C2)** Relational Database to RDF Graph (RDB2RDF) Mapping Engineering: defining mappings from the complex relational database schemas to the ontology. This is probably not surprising to a semantic web reader. However, what has been surprising to us is the ease of how one can fall into a "boiling the ocean" state when trying to combine these two tasks. This is evident when dealing with complex real-world enterprise relational database schemas.

(C1): Ontology Engineering Engineering ontologies is difficult in and of itself. The field of Ontology Engineering was prolific throughout the 90s. Early seminal work by Fox, Gruninger, King and Uschold pioneered the field of ontology engineering [15, 16, 3], followed by a multitude of methodologies, notably METHONTOLOGY [2]. Research in this field continues to progress by focusing on the sophisticated use of competency questions [1, 8], test-driven development [6], ontology design patterns [4], reuse [9], etc to name a few. Furthermore, numerous ontologies have been designed so that they can be reused, such as Good Relations for e-commerce, FIBO for finance, Schema.org, and so on.

It seems to be a fair conjecture that C1 should actually not be challenge. This would be the case if the ultimate deliverable is just an ontology in isolation. However, populating an ontology with instances coming from a relational database, seems to be an afterthought and not a key component of existing ontology engineering methodologies. In the context of designing a Knowledge Graph from relational databases, both the ontology and the RDB2RDF mappings must be first-class citizens.

(C2): RDB2RDF Mapping Engineering Let's assume that an ontology has been created via an established methodology. The next step is to map relational databases to the ontology in order to generate the graph data. A common theoretical practice is to bootstraps with the automatic direct mapping that generates a so-called *putative ontology* from the database schema [13]. This practice suggests approaching the problem as an ontology-matching problem between the putative ontology and the target ontology. In theory, this can work [5].

However, per our real-world experience, this has not yet (and may never) become practicable in the real-world for the following reasons: (1) Commonly, enterprise relational database schema are very large, consisting of thousands of tables and tens of thousands of attributes (Oracle EBS has +20,000 tables!). Schema developers notoriously use peculiar abbreviations which are meaningless. Commercial systems make frequent use of numbered columns with no explicit semantics (e.g. segment1, segment2, etc). (2) Simple one-to-one correspondence between table-classes and columns-properties are rare. In all of our commercial deployments, complex mappings dominate: mappings with calculations, business logic that simultaneously considering database values. For example, the mapping to the target property *net sales* of a class *Order* is defined as gross sales minus taxes and discounts given. Tax rates can be different depending on location. Discounts can depend on the type of customer. A business user needs to provide these definitions beforehand. Thus, without clairvoyance, automating mapping is often simply not plausible. (3) It is not plausible that we will ever have copious amounts of schema to successfully train machine learning algorithms.

Early on in our practice we observed that ontologies and mappings must be developed holistically. That is, there is a continual back-and-forth between ontology and mapping engineering. Thus, ontology engineering methodologies must be extended to support RDB2RDF mapping engineering.

3 The Pay-as-you-go Methodology

Our proposed Pay-as-you-go Methodology address the two aforementioned challenges by combining ontology and RDB2RDF mapping engineering. At the center of the methodology are a set of prioritized business questions that need to be answered. The business questions serve as competency questions and as a success metric.

In the initial knowledge capture phase, the first business question is analyzed, understood, modeled into a minimal viable ontology, and mapped to a database. In the following knowledge implementation phase, the ontology and mappings are implemented in OWL and R2RML respectively. The resulting data can be (virtually or physically) accessed, validated and imported into one or more BI tools. In the final self service analytics phase the BI dashboards are used to answer the initial business questions. Once this initial iteration has occurred, the next business question is analyzed. If the next question can be answered with the current ontology and mappings, then we are done. Otherwise, the ontology is extended incrementally with its corresponding mappings. With this approach, the ontology and mappings are developed simultaneously in an agile and iterative manner: hence, pay-as-you-go.

The ontology expressivity we focus on is owl:Class, owl:DatatypeProperty (with domain and range), owl:ObjectProperty (with domain and range) and rdfs:subClassOf, thus RDFS without subproperties. We have found out that this expressivity is sufficient to address the BI needs for our customers. Furthermore, we have identified that the following terminology is well received: Concept rather than Class. Attribute replaces Data Property. Relationship is used in place of Object Property. We colloquially refer to Concepts, Attributes and Relationships as CARs.

This methodology was developed and refined over a number of customers, throughout the last 4 years. Furthermore, it builds upon the extensive work in ontology engineering over the past two decades. For example, common steps across all methodologies is to identify a purpose, define competency questions and formalize the terminology in an ontology language. Specifically, we are inspired and extend the notion of Intermediate Representations (IRs) from METHONTOLOGY [2].

We identify three actors involved throughout the process: **(1) Business Users** are subject matter experts who can identify the list of prioritized business questions, understand the business rules associated with the data and validate the integrity of the created data. **(2) IT Developers** understand database schemas, including how the data are interconnected. **(3) Knowledge Scientist**[8] serve as the communication bridge between Business Users and IT Developers.

The methodology is organized in three phases, with different expectations from each actor throughout the process: **(Phase 1) Knowledge Capture:** the Knowledge Scientist works with the Business User to understand the business questions, define an "whiteboard" version of the ontology and work with the IT Developer to determine which data is needed. This is documented in a knowledge report. **(Phase 2) Knowledge Implementation:** the Knowledge Scientist can now implement the ontology, mappings and queries based on the content from the knowledge report. **(Phase 3) Self-Service Analytics:** the Business User is now exposed to the data in a simplified and easy to understand view enabling straightforward data access with common BI tools. They can now create reports and dashboards to provide answers to new and existing business questions without having to further interface with IT.

Phase 1: Knowledge Capture The knowledge capture phase must accomplish two objectives: 1) Users must understand and clarify the business questions, and 2) Users must identify the data necessary to answer those business questions.

Step 1: Analyze Processes: The goal is to analyze and formalize existing processes because many these processes may have never been written down before. When a business question needs to be answered, we first need to understand the larger context: *what is the business problem that needs to be addressed? Is it currently being addressed, and if so, how?* Answering the following questions help achieve this goal:

What: What are the business questions? What is the business problem?

Why: Why do we need to answer these questions? What is the motivation?

Who: Who produces the data? Who will consume the data? Who is involved?

How: How is this the business question answered today, if at all?

Where: Where are the data sources required to answer the business questions?

When: When will the data be consumed? Real-time? Daily? Update criteria?

Step 2: Collect Documentation: In this step, the Knowledge Scientist focuses on the answers to the ***How*** and ***Where*** questions from the previous step. They identify documentation about the data sources and any SQL queries, spreadsheets, or scripts being used to answer the business questions today. They may also interview stakeholders to understand their current workflow.

Step 3: Develop Knowledge Report: The Knowledge Scientist analyzes what was learned in steps 1 and 2 and starts working with the Business User to understand the business questions, recognize key concepts and relationships from the business questions, identify the business terminology such as preferred labels, alternative labels, and natural

[8] Traditionally, this role has been called Knowledge Engineer. We decided to call it Knowledge Scientist so it can be on-par with the title of the "Data Scientist".

language definitions for the concepts and relationships. At this stage, it is common to encounter disagreements. Different people use the same word to mean different concepts or different words are used to mean the same concept. The conversation is very focused on the business questions which helps drive to a consensus. Subsequently, the

Concept Name	The agreed name for the Concept
Concept Definition	The agreed definition for the Concept
Concept ID	The ID that will uniquely identify the Concept and will from a URI
Unique ID of a Concept	The attribute from the Table Name/SQL query that uniquely identifies each instance of the Concept
Table Name or SQL Query	The Table Name or SQL query logic that represents the Concept

Table 1. Knowledge Report: Concepts

Knowledge Scientist works with the IT developer to identify which tables and attributes in the database contains data related to the concepts and relationships identified from the business questions. The conversation with IT is also focused.

An outcome of this step is a high-level view of the ontology– a whiteboard illustration. The final deliverable is a knowledge report which can be conveniently represented as a spreadsheet consisting of multiple tabs detailing the CARs (Concepts, Attributes and Relationships), with the corresponding SQL logic which serve as the mapping to the relational database. Subsequently there is a tab for each Extract, which is a definition of the tabular result that a Business User would like to have access. Each row in an Extract tab should list all the Attributes that will appear in the extract. The template for the knowledge report is shown in Table 1, 2 and 3.

The diligent reader will notice that our notion of Knowledge Reports mimics the Intermediate Representations (IRs) from METHONTOLOGY. In the METHONTOLOGY conceptualization phase, the informal view of a domain is represented in a semi-formal specification using IRs which can be represented in a tabular or graph representation. The IRs can be understood by both the Domain Experts and the Knowledge Scientist, therefore bridging the gap between the business users informal understanding of the domain and the formal ontology language used to represent the domain. The Knowledge Scientist will report back to the Business User and IT developer and explain, using the knowledge report, how the business concepts are related and how they are connected to the data. If all parties are in agreement, then we can proceed to the next phases. Otherwise, the discrepancies must be resolved. These can be identified quickly due to the granularity of the knowledge report.

Phase 2: Knowledge Implementation A key insight of METHONTOLOGY's Intermediate Representations, is that they ease the transformation into a formal ontology language. We build upon this insight. That is why the goal of the knowledge implementation phase is to formalize the content of the knowledge report into an OWL ontology, R2RML mappings, SPARQL queries and subsequently validate the data. As noted below, the different elements of the Knowledge Report has its correspondences to OWL and R2RML.

Attribute Name	The agreed name for the Attribute
Attribute Definition	The agreed definition for the Attribute
Attribute ID	The ID that will uniquely identify the Attribute and will from a URI
Applied to Concept	The Concept for which this attribute is associated to.
Unique ID of a Concept	The attribute from the Table Name/SQL query that uniquely identifies each instance of the Concept
Table Name or SQL Query	The Table Name or SQL query logic that represents the Concept
Datatype	The expected datatype of the Attribute
Is NULL possible?	Can there be NULL values in the column name? Yes or No.
If NULL?	If the column can have NULLs, then what is the default value ("NULL", "N/A", 0, etc)

Table 2. Knowledge Report: Attribute

Step 4: Create/Extend Ontology: Based on the knowledge report, the Knowledge Scientist can create the ontology or extend an existing ontology. Table 4 details the correspondence between the elements of the knowledge report and constructs of OWL in order to create an OWL ontology using any ontology editor.

Relationship Name	The agreed name for the Relationship
Relationship Definition	The agreed definition for the Relationship
Relationship ID	The ID that will uniquely identify the Relationship and will from a URI
From Concept	What Concept does this relationship come from (the domain)
Unique ID of From Concept	The attribute from the Table Name/SQL query that uniquely identifies the From Concept
Table Name or SQL Query	The Table Name or SQL query logic that returns the data for the Relationship. This query usually returns a pair of attributes which includes the IDs of the From and To Concept
To Concept	To what Concept does this relationship connect to (the range)
Unique ID of To Concept	The attribute from the Table Name/SQL query that uniquely identifies the To Concept

Table 3. Knowledge Report: Relationships

Step 5: Implement Mapping: Similar to the previous step, the Knowledge Scientist can create the R2RML mappings from the knowledge report. Table 5 details the correspondence between the elements of the knowledge report and constructs of R2RML in order to create an R2RML mapping.

Step 6: Extract Queries : Recall that an Extract is the definition of the tabular result that a Business User would like to access. A SPARQL SELECT query is the implementation of the Extract. The SPARQL query is executed in an OBDA system, such as Ultrawrap, using the R2RML mapping from the previous step.

Knowledge Report	**OWL Ontology**
Concept	owl:Class
Concept Name	rdfs:label of the Class
Concept Definition	rdfs:comment of the Class
Concept ID	used to create the URI of the Class
Attribute	owl:DatatypeProperty
Attribute Name	rdfs:label of the Datatype Property
Attribute Definition	rdfs:comment of the Datatype Property
Attribute ID	used to create the URI of the Datatype Property
Applied to Concept	rdfs:domain of the Datatype Property
Datatype	rdfs:range of the Datatype Property
Relationship	owl:ObjectProperty
Relationship Name	rdfs:label of the Object Property
Relationship Definition	rdfs:comment of the Object Property
Relationship ID	used to create the URI of the Object Property
From Concept	rdfs:domain of the Object Property
To Concept	rdfs:range of the Object Property

Table 4. Correspondence between Knowledge Report and OWL constructs

Step 7: Validate Data: The final step is to validate the extracted data, resulting from the SPARQL query in the previous step. This data should also be validated by the business users. Suggested validation techniques are the following: sharing sample data to business users in a spreadsheet, creating sample visualizations in a BI tool, comparing the number of results in the extract with the source database to ensure they are the same, and checking the validity of NULL values.

After successful data validation with the business users, the data can begin to be used for Self-Service Analytics in the next phase. Otherwise, the root cause of the error must be found. This commonly in either in the SPARQL query or an R2RML mapping.

Phase 3: Self Service Analytics

Step 8: Build Report: A key component of Self-Service Analytics is for business users to use BI tools over a simplified view of the data. The ontology enables the simplified view but at the end, it is the tabular extract that the business user wants to access. Through our experience, the primary ways of BI tools access the extract are through uploads of csv files, special connectors developed for SPARQL or caching/materializing the result of a SPARQL query in a relational database. Once the extract(s) is available in the BI tool, the business user is enabled to build their business intelligence report.

Step 9: Answer Question: The BI report should answer the original business question (the **What** in Step 1). This report is shared with the stakeholders who asked the original business question (the **Who** in Step 1). If they accept the BI report as an answer to their question, then this is ready to move to production.

Step 10: Move to production: Once the decision has been made to move to production, we need to determine how the BI tools are going to access the extracts. If it is going to be through a SPARQL connector then it can be done live via the SPARQL endpoint. If the data is going to be accessed through a cache, the refresh schedule must be determined. Common refresh schedules are daily, weekly, monthly or on demand. Additionally, the time window of the extract needs to be determined. Is the cache going to update the entire extract, or is only yesterday's data going to be pushed to the cache? or last weeks? These questions must be answered before wide release.

Knowledge Report	**R2RML Mapping**
Concept: Concept ID	mapping to URI of the class in rr:Class
Concept: Unique ID of a Concept	rr:template
Concept:Table Name or SQL Query	rr:logicalTable
Attribute: Attribute ID	mapping to URI of the class in rr:predicate
Attribute: Unique ID of a Concept	rr:template
Attribute: Table Name or SQL Query	rr:logicalTable
Attribute: Column Name	rr:column
Relationship: Relationship ID	mapping to URI of the class in rr:predicate
Relationship: Unique ID of From Concept	rr:template
Relationship: Table Name or SQL Query	rr:logicalTable
Relationship: To Concept	rr:joinCondition

Table 5. Correspondence between Knowledge Report and R2RML constructs

4 An E-Commerce Case Study

Background: This case study is based on a Capsenta customer which is an e-commerce company selling health and beauty products in about a dozen countries. As in any large enterprise, BI is a key driver to help make critical business decisions. The customer had invested large amounts of time and money in an enterprise data warehouse (EDW) that integrated data coming from several of their multiple heterogeneous databases. The business users were skeptical of BI reports coming from the existing EDW. When comparing results between the EDW and those coming directly from the original source databases, the results were effectively different. Due to the lack of trust in the EDW's data, the business users decided to move off the EDW and start building BI reports by pulling data from the original source databases.

Challenge: To answer the business questions, the business users needed to create reports which required data from multiple, disparate, mostly inscrutable relational database schemas. Business users would request data from IT and in return they would receive the results, typically, in a csv file. A small number of business users had sufficient SQL knowledge to extend SQL queries that were given to them from IT. However, given the complexity of the source relational schemas, these business users still needed to request help from IT. There was friction between the business users and IT due to lack of agreement on terminology and frequently the business users would not get the data they expected. Thus, IT was considered a bottleneck in delivering answers to business questions. When data coming from different databases needed to be combined, it would be ad-hoc merged locally either in Excel or in a MS Access database. Even though the business users had direct access to the multiple databases, which gave a feeling of ownership and control, the results were inconsistent and, hence, not trusted. When C-Level executives would ask business questions, multiple diverging answers were being provided because the business logic used to deliver the BI reports were severally and inconsistently implemented. Suffice it to say that the process of generating BI reports was ad-hoc, error-prone, time consuming and unscalable.

Furthermore, the company was in the process of bringing on board Tableau as a new, corporate-wide BI solution and IT executives feared that the current ad-hoc process could be detrimental to the implementation of Tableau and potentially it could suffer the same fate as the EDW.

Solution: The customer needed a consistent, understandable and trusted data view across the multiple relational databases such that BI tools (e.g., Tableau) could consume the data and a large number of business users could generate reports all using the same trusted data. Furthermore, the customer required an agile approach in order to start showing value quickly. Unsatisfied with the status quo of large scale enterprise data warehouse projects and not wanting to go down the same route again, the customer started looking into semantic technology.

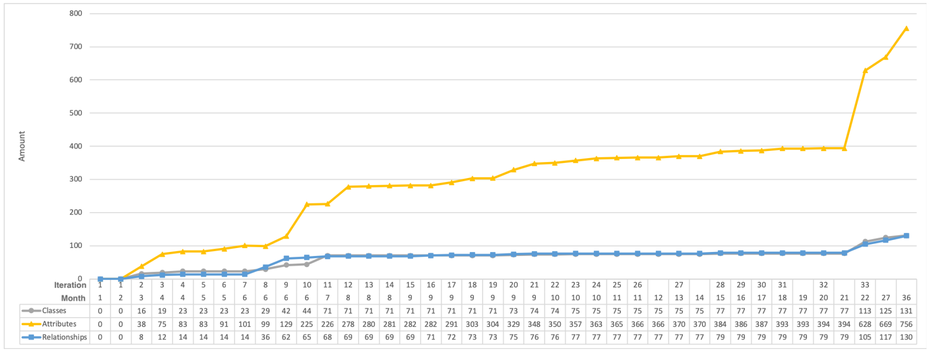
The solution was to create an enterprise ontology which would then be mapped to the different relational databases. The OWL enterprise ontology would serve as the business user's common view and lingua franca. The R2RML mappings would ground the meaning of the business conceptualizations with data, thus serving as a means of communication between business and IT users. The business questions, represented in SPARQL queries, would be in the user-friendly terms of the enterprise ontology. Capsenta's Ultrawrap product family would be the OBDA platform that would tie together the ontology, mappings and queries. The results of SPARQL queries would feed different BI tools. The pay-as-you-go methodology was required in order to provide an agile development process which could quickly show the C-level executives that this wasn't going to be another multi-month, million dollar failure (i.e., their last EDW).

Results: In order to show the effectiveness of the technology and kick off the first iteration of the methodology, the goal was to replicate the most trusted BI report: the daily sales report that all C-level executives viewed every morning. There were three Business Users, two IT users and one Knowledge Scientist involved.

The daily sales report was being generated by one SQL query that a business user would execute every morning. The size of SQL query, in text, was 21KB. Needless to say, this was a large SQL query. During the Knowledge Capture phase, we learned that the business user's SQL query was an extension of another SQL query created initially by an IT user. We observed multiple discrepancies between the Business and the IT user's SQL queries. The Knowledge Capture phase revealed that the daily sales report encompassed 16 concepts, 38 attributes and 8 relationships. Per the methodology, each CAR has a corresponding mapping. The customer was surprised to see how much knowledge was "hidden" within just one report.

Fig 1 depicts how the CARs evolved over 36 months and 33 iterations. Capsenta worked directly with the customer for the first 22 months. Afterwards, the customer took full control of the project, therefore we did not track all the iterations after month 22. The customer continues to deploy Ultrawrap and apply this methodology today. The following phases occurred.

1) Rapid Growth Phase (Month 1 - 7): The first iteration took 2 months due to an initial ramp up of the project. The following 10 iterations took 5 months (approx one



Iteration	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Month	1	2	3	4	4	5	5	6	6	6	6	7	8	8	8	9	9	9	9	9
Classes	0	0	16	19	23	23	23	23	29	42	44	71	71	71	71	71	71	71	71	71
Attributes	0	0	38	75	83	83	91	101	99	129	225	226	278	280	281	282	282	291	303	304
Relationships	0	0	8	12	14	14	14	14	36	62	65	68	69	69	69	69	71	72	73	73

Iteration	20	21	22	23	24	25	26		27		28	29	30	31		32		33		
Month	9	9	10	10	10	11	11	12	13	14	15	16	17	18	19	20	21	22	27	36
Classes	73	74	74	75	75	75	75	75	75	75	77	77	77	77	77	77	77	113	125	131
Attributes	329	348	350	357	363	365	366	366	370	370	384	386	387	393	393	394	394	628	669	756
Relationships	75	76	76	77	77	77	77	77	77	77	79	79	79	79	79	79	79	105	117	130

Fig. 1. Analysis of iterations by month

iteration every 2 weeks). During this phase, the ontology and mappings grew quickly because we were understanding the most important business concepts and how they were interrelated. The focus was to understand a proprietary order management system which consisted of three databases.

2) Consistent Growth Phase (Month 8-22): 21 iterations occurred within the following 14 months (approx one iteration every 2.5 weeks too). The growth in the stage was consistent because, the focus was on adding attributes instead of concepts and relationships. The relational database underlying Oracle E-business Suite was integrated.

3) Independent Growth Phase (Month 23-present): The customer was able to independently follow the methodology and continue to grow the ontology and mappings. As previously mentioned, we do no have data about the iterations during this time. Three additional relational databases were integrated: event management system, shipping database and a miscellaneous database that hosts spreadsheets.

Further analysis can be made from this data: **(A)** Concepts and relationships increased at the same rate because every time a concept is added, it is connected to another concept, hence a relationship is also added. On average, 2.2 concepts and relationships were added per iteration while the median is 0. The max number of concepts and relationships added in an iteration was 27 while the min was 0. **(B)** Attributes grew the most because this is where the actual data values from the source databases is coming from. On average, 11.2 attributes were added per iteration while the median is 2. The max number of attributes added in an iteration was 96 while the min was -2 (two attributes were eliminated). **(C)** The busiest month was Month 9 which had 7 iterations (an iteration every couple of days) focused on incrementing attributes. 66 new attributes were added. **(D)** Months 12, 14, 19 and 21 did not have any iterations because there was no new requests from business users, employees were on vacation, etc.

In every iteration, at minimum there was at least one Business User, Knowledge Scientist and IT user. At peak, there were three Business Users, three Knowledge Scientist and two IT users.

Before the start of this project (over 3 years ago), there were only a handful of BI developers. Today there are +20 active BI developers creating reports in Tableau. +100 business users are consuming these reports every day. Due to confidential reasons, we cannot provide details about the business questions that are being answered.

To further exemplify how the Pay-as-you-go Methodology is applied, we provide a real-world in-use example from the case study.

Round 1: Orders

Phase 1: Knowledge Capture We start by asking the questions in Step 1 above, and fill in the appropriate answers:

What: How many orders were placed in a given time period per their status?

Why: Depending on whom is asked, different answers can be provided. Unaware of the source of the problem, the executives are vexed by inconsistencies across established business reports.

Who: The Finance department, specifically the CFO

How: A business analyst asks the IT developer for this information every morning.

Where: There is a proprietary Order Management System and Oracle E-Business Suites.

When: Every morning they want to know this number

The Knowledge Scientist gathers access to the database systems for the Order Management System and the Accounting System and learns that the Order Management System was built on an open-source shopping cart system and has been heavily customized. It has been extended repeatedly over the past years and the original architect is no longer with the company. Documentation about the database schema does not correspond to the production database schema. Furthermore, the database schema of the Order Management System consists of thousands of tables and 10 tables have the word “order” in the name with different types of prefixes (masterorder) and suffixes (ordertax). Finally, the Knowledge Scientist gets the SQL script that the IT developer runs every morning to generate the data that is then passed along to a business analyst. Also, the IT developer did not write this SQL script; it was passed to them from previous employees.

The Knowledge Scientist works with the Business User to understand the meaning of the word “order.” Discussions with the business revealed that the definition of an *order* is if it had shipped or the accounts receivable had been received.

Together with the IT developer, the Knowledge Scientist learns that the Order Management System is the authoritative source for all orders. Within that database, the data relating to orders is vertically partitioned across several tables. The SQL scripts collected in the previous step provides focus to identify the candidate tables and attributes where the data is located. Only the following tables and attributes are needed from the thousands of tables and tens of thousands of attributes: `MasterOrder(moid, oid, masterdate, ordertype, osid, ...)`, `Order(oid, orderdate, ...)` , `OrderStatus(osid, moid, orderstatusdate, ostid, ...)`, `OrderStatusType (ostid, statustype, ...)` .

Together, they identify the business requirement of an order as all tuples in the MasterOrder table, where the ordertype is equal to 2 or 3. Note that in some SQL scripts, this condition was not present. This is the reason why the Finance department was getting different answers for the same question.

Furthermore, it is revealed that the table OrderStatus holds all the different status that an order has across different periods of time. In discussions with the business user, it is confirmed that they only want to consider the last order status (they do not care about the historic order statuses). This may have been another source of differing numbers because a single order can have multiple order statuses, but it is unique for a given period a time.

Concept Name	Order
Concept ID	Order
Unique ID of a Concept	moid
Table Name or SQL Query	select moid from masterorder m join order o on m.oid = o.oid where ordertype in (2,3)

Table 6. Concept Knowledge Report for Round 1

The Knowledge Report is in Table 6 - 8. Due to space limitation, a single CAR is shown. The extract consists of Order Number, Order Date and Order Status.
Phase 2: Knowledge Implementation The ontology is the following:

```
ec:Order rdf:type owl:Class ; rdfs:label "Order" .
ec:orderDate rdf:type owl:DatatypeProperty ; rdfs:domain ec:Order ;
  rdfs:range xsd:dateTime ; rdfs:label "Order Date" .
 ec:hasOrderStatus rdf:type owl:ObjectProperty ; rdfs:label "has order status"
  rdfs:domain ec:Order ; rdfs:range ec:OrderStatus .
```

The following is an example R2RML mapping for the concept Order:

```
map:m1 a rr:TriplesMap ;
  rr:logicalTable  [ rr:sqlQuery  "select moid from masterorder m join
                            order o on m.oid = o.oid where ordertype in (2,3)" ] ;
   rr:subjectMap    [ rr:class     ec:Order ;
     rr:template  "http://www.e-commerce.com/data/Order/{moid} ] .
```

The SPARQL query to generate the extract is the following:

```
SELECT ?Order_Number ?Order_Date ?Order_Status
WHERE {
?x a :Order;
  :orderNumber ?Order_Number;
  :orderDate ?Order_Date;
  :hasOrderStatus [
     :orderStatusName ?Order_Status;
  ]
}
```

Sample data is provided to the business users and IT developer and validated.
Phase 3: Self-Service Analytics The extract is now accessible in a BI tool. The Business User sees a very simple table with the accurate data that is needed in order to build the business intelligence report and answer the original query: “How many orders were placed in a given time period per their status?”

This data is now accessible to a large number of business users which before would have had to talk to IT in order to get the same or similar data. Assume the extract was cached in a table called `Orders`. Any business user can retrieve all the data about Orders by simply executing: `SELECT * FROM Orders`.

Attribute Name	Order Date
Attribute ID	orderDate
Applied to Concept	Order
Unique ID of a Concept	moid
Table Name or SQL Query	select moid, orderdate from masterorder m join order o where m.oid = o.id
Column Name	orderdate
Datatype	date
Is NULL possible?	No
If NULL?	N/A

Table 7. Attribute Knowledge Report for Round 1

Relationship Name	has order status
Relationship Definition	An order has a Order Status
Relationship ID	hasOrderStatus
From Concept	Order
Unique ID of From Concept	moid
Table Name or SQL Query	select moid, ostid, max(orderstatusdate) from OrderStatus group by orderstatusdate
To Concept	Order Status
Unique ID of To Concept	ostid

Table 8. Relationship Knowledge Report for Round 1

To get the exact same data directly from the database of the Order Management System, the business user would have to spend time with IT to determine the SQL query, which would have been:

```
SELECT m.moid as OrderNumber, o.orderdate as OrderDate,
        ost.statustype as OrderStatusName
FROM masterorder m
JOIN order o ON m.oid = o.oid
JOIN (SELECT moid, ostid, max(orderstatusdate)
     FROM OrderStatus GROUP BY orderstatusdate) os ON m.moid = os.moid
JOIN OrderStatusType ost ON os.ostid = ostid.ostid
WHERE m.ordertype in (2,3)
```

Round 2: Order Net Sales In order to show the pay-as-you-go nature of the methodology, consider the following new request:
Phase 1: Knowlege Capture
What: What is the net sales of an order?
Why: Depending on whom is asked, different answers are provided. The net sales is dependent on at least 4 different aspects of each order and sometimes aspects of each individual line item. The departments and individuals reporting results are variously not applying all of the proper items, not applying them consistently or not applying them correctly (per the business' desired rules).
Who: The Finance department, specifically the CFO
How: A business analyst asks the IT developer for this information every morning.
Where: This is in the proprietary Order Management System.
When: Every morning they want to know the net sales of every order and also various statistics and aggregations.

In conversations with the Business User, the Knowledge Scientist learns that the business user gets a CSV file from IT. The business user opens it in Excel and applies some calculations. The Knowledge Scientist works with the business user to understand the meaning of the word “order net sales”. It is then understood that the net sales of an order is calculated by subtracting the tax and the shipping cost from the final price and also adjusting based upon the discount given. However, if the currency of the order is not in USD or CAD, then the shipping tax must be subtracted. Working with IT, they identify another table that is needed: `ordertax`. It is noted that the ontology only needs to be extended to support two new Attributes: Order Net Sales and Order Currency as shown in Table 9. Finally the original Extract is extended with the two new attributes.

Attribute Name	Order Net Sales
Attribute ID	orderNetSales
Applied to Concept	Order
Unique ID of a Concept	moid
Table Name or SQL Query	select moid, o.ordertotal - ot.finaltax - CASE WHEN o.currencyid in (“USD”, “CAD”) THEN o.shippingcost ELSE o.shippingcost = ot.shippingtax END as ordernetsales from masterorder m join order o on m.oid = o.id join ordertax ot on o.oid = ot.oid
Column Name	netsales
Datatype	float
Is NULL possible?	No
If NULL?	N/A

Table 9. Attribute Knowledge Report for Round 2

Phase 2: Knowledge Implementation The existing ontology has now been extended with the following,:

```
ec:orderNetSales rdf:type owl:DatatypeProperty ; rdfs:domain ec:Order ;
  rdfs:range xsd:float ; rdfs:label "Order Net Sales" .
ec:orderCurrency rdf:type owl:DatatypeProperty ; rdfs:domain ec:Order ;
  rdfs:range xsd:string ; rdfs:label "Order Currency" .
```

The following is R2RML mapping for Order Net Sales

```
map:m3 a   rr:TriplesMap ;
  rr:logicalTable [ rr:sqlQuery  "select moid, o.ordertotal - ot.finaltax -
      CASE WHEN o.currencyid in (?USD’, ?CAD) THEN o.shippingcost
      ELSE o.shippingcost = ot.shippingtax END as ordernetsales FROM ... " ] ;
  rr:predicateObjectMap  [ rr:objectMap  [ rr:column  "ordernetsales" ] ;
                             rr:predicate  ec:orderNetSales ] ;
  rr:subjectMap [ rr:template  "http://www.e-commerce.com/data/Order/{moid}" ] .
```

The existing SPARQL query is extended as follows:

```
SELECT ?Order_Number ?Order_Date ?Order_Status ?Order_Net_Sales ?Order_Currency
WHERE {
?x a :Order;
  :orderNumber ?Order_Number;
  :orderDate ?Order_Date;
  :orderNetSales ?Order_Net_Sales;
  :orderCurrency ?Order_Currency;
  :hasOrderStatus [
     :orderStatusLabel ?Order_Status;
  ]
}
```

Phase 3: Self-Service Analytics With the extended extract, the business users can further enhance the business intelligence report in order to answer the new question of this round.

5 Lessons Learned, Future Work and Conclusions

Knowledge Scientist: Success with the Pay-as-you-go Methodology depends on having a Knowledge Scientist– someone with a broad set of technical and social skills. Such people may be hard to find. We have learned that potential candidates have a technical background in data (SQL developers, etc.), know data modeling (UML, etc.), enjoy creating documentation, and commonly interact with business users. If looking internally, they are employees who have been at the organization for a long time and understand how the business functions. Potential candidates have dual backgrounds in computer science and arts (literature, music, etc). Furthermore, the Knowledge Scientist should not be the Data Scientist. The Knowledge Scientist serves as a communication bridge between business and IT to understand the data. The Data Scientist works with the business to generate new insights and analysis with the data generated by the Knowledge Scientist.

Snowball Effect: It can sometimes take a while to get the ball rolling but once business users see the first bits of understandable data they get excited and want more. End users are empowered to ask questions that they had not even considered before with the status quo process. The Pay-as-you-go Methodology enables quickly adding more data in a way that business users can understand and easily access. The business benefit can be seen quickly and expanded. This feeds still more excitement for more data. The snowball gets larger and increasingly rolls down the hill faster. Additionally, the CIO can see tangible, early success (as opposed to the typical EDW) and feels comfortable funding the activities or expanding funding.

Manual Process: The amount of manual work and iterations could be seen as inefficient and expensive. We acknowledge this limitation. Regardless of the repetitions, we observe that when users understand the methodology, the iterations are faster. For example, the first iteration took 2 months, later on 4 iterations were done in one month. The repetitive process makes us reflect: what parts of the of methodology can we (semi-)automate? We find this a challenging problem because modeling can be seen as much of a science as it is an art.

Maintenance and Evolution: The ontology and mappings were created in an additive manner. In very rare cases we had to go back and make changes (in month 8, two attributes were eliminated). We attribute this to the fact that in the methodology clearly defines stages where there has to be reviews. If something is not clear from the beginning, we do not go further next step. We did not address the phenomena of schema changes and ontology evolutions. How should the methodology be adopted when there are database schema changes or if the ontology is updated outside of the methodology? What happens if these changes are monotonic or non-monotonic?

Ontology Expressivity: The ontologies that are generated with the Pay-as-you-go Methodology have a basic expressivity. One can consider that the limited expressivity is a limitation. In our case it was an advantage. For non-semantic aware customer, simple and less complexity reduced the barrier to entry. This has proven to be sufficient for the BI tasks of our customers. We acknowledge that this is not a methodology to create profoundly expressive ontologies. We believe there is opportunity to extend the methodology to support the engineering of more expressive ontologies.
Evaluation: In this paper, we presented a case study that shows 33 iterations over three years. Is this good? Is this bad? It is hard to provide an objective answer to this question. We believe it is critical to devise rigorous and thorough scientific evaluation methodologies for knowledge graph construction methodologies.
Tools: Existing tools have been designed for users with a strong background in semantic technologies. A clear need exists for ontology and mapping tools designed for non-semantic aware users. Furthermore, the tools need to be aligned with methodologies. Capsenta started out by developing Ultrawrap Mapper [14]. Subsequently, we developed Gra.fo (`https://gra.fo`), a visual, collaborative and real time knowledge graph schema/ontology editor. Both of these tools are in being integrated so they support this methodology. We believe it is paramount that tools are designed in conjunction with a methodology. We continue to refine the tools and techniques to be maximally useful to our non-semantic users while also exposing the much larger general business user and IT world to the valuable capabilities the semantic community has delivered.

At Capsenta, we now use the Pay-as-you-go Methodology, Gra.fo and Ultrawrap [11, 14] in all customer engagements.

References

1. Azzaoui et al., Scientific Competency Questions as the Basis for Semantically Enriched Open Pharmacological Space Development Drug Discovery Today, vol. 18, no. 17-18, 2013.
2. Fernandez-Lopez et al., METHONTOLOGY: From Ontological Art Towards Ontological Engineering, AAAI Symposium on Ontological Engineering, 1997.
3. Gruninger & Fox. Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.
4. Hitzler et al., eds., Ontology Engineering with Ontology Design Patterns: Foundations and Applications. Studies on the Semantic Web 25, IOS Press/AKA, 2016.
5. Jimenez-Ruiz et. al. BootOX: Practical Mapping of RDBs to OWL 2. ISWC 2015
6. Keet and Lawrynowicz. Test- Driven Development of Ontologies. ESWC 2016
7. Kharlamov et. al. Ontology Based Data Access in Statoil. JWS 2017
8. Ren et al. Towards Competency Question-Driven Ontology Authoring. ESWC 2014
9. Suárez-Figueroa eds., Ontology Engineering in a Networked World. Springer. 2012.
10. Sequeda et.al. OBDA: Query Rewriting or Materialization? In Practice, Both! ISWC 2014
11. Sequeda & Miranker. Ultrawrap: SPARQL Execution on Relational Data. JWS 2013.
12. Sequeda & Miranker. A Pay-As-You-Go Methodology for Ontology-Based Data Access. IEEE Internet Computing 21(2). 2017
13. Sequeda et.al. On Directly Mapping Relational Databases to RDF and OWL. WWW2012
14. Sequeda and Miranker. Ultrawrap Mapper: A Semi-Automatic Relational Database to RDF (RDB2RDF) Mapping Tool. ISWC Posters & Demos, 2015
15. Uschold & Gruninger. Ontologies: Principles, Methods and Applications. KER 1996.
16. Uschold & King. Towards a Methodology for Building Ontologies, in: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.