The Credit Suisse Meta-data Warehouse

Claudio Jossen #1, Lukas Blunschi #2, Magdalini Mori #3, Donald Kossmann #4, Kurt Stockinger #5

# Credit Suisse AG
Switzerland
1 claudio.r.jossen@credit-suisse.com
2 lukas.blunschi@inf.ethz.ch
3 magdalini.mori@credit-suisse.com
4 donald.kossmann@inf.ethz.ch
5 kurt.stockinger@credit-suisse.com

# ETH Zurich
Switzerland

Abstract—This paper describes the meta-data warehouse of Credit Suisse that is productive since 2009. Like most other large organizations, Credit Suisse has a complex application landscape and several data warehouses in order to meet the information needs of its users. The problem addressed by the meta-data warehouse is to increase the agility and flexibility of the organization with regards to changes such as the development of a new business process, a new business analytics report, or the implementation of a new regulatory requirement. The meta-data warehouse supports these changes by providing services to search for information items in the data warehouses and to extract the lineage of information items. One difficulty in the design of such a meta-data warehouse is that there is no standard or well-known meta-data model that can be used to support such search services. Instead, the meta-data structures need to be flexible themselves and evolve with the changing IT landscape. This paper describes the current data structures and implementation of the Credit Suisse meta-data warehouse and shows how its services help to increase the flexibility of the whole organization. A series of example meta-data structures, use cases, and screenshots are given in order to illustrate the concepts used and the lessons learned based on feedback of real business and IT users within Credit Suisse.

I. INTRODUCTION

A large organization like Credit Suisse [1] has thousands of applications that support different parts of the business. Even though there are efforts to consolidate, the number of applications keeps growing. Furthermore, each application creates an increasing number of features with every release as the requirements and information needs of business users continue to grow or change. At the same time, the IT infrastructure that is needed in order to maintain these applications and deliver the information needs of the business is quite dynamic; new technologies such as service-oriented architectures and cloud computing have significantly changed the way organizations deploy software and hardware.

Traditionally, data warehouses have provided an abstraction layer that made it possible to fulfill the information needs of many business users. A data warehouse integrates data from multiple applications and provides a uniform data model to access that data. Furthermore, a data warehouse hides the complexity of the IT infrastructure, thereby following the principle of data independence.

While data warehouses are a powerful and indispensable tool, they are not sufficient. Due to the increasing and dynamically changing business needs, data warehouses have grown in complexity and have become (almost) as difficult to manage as the applications and data sources that they integrate. Specifically, the schemata of data warehouses have grown organically over the years. As a result, there is no single user anymore who can oversee and understand all tables of a data warehouse. Furthermore, the same data may be replicated several times in a data warehouse, thereby giving access to data in different granularity, freshness, and quality. Most modern data warehouses such as those deployed at Credit Suisse are organized in different layers (e.g., staging areas or enrichment areas) in order to provide a data cleansing and aggregation pipeline. All stages of this pipeline are needed and used by business users as they all provide different freshness, response time, and data quality guarantees.

Clearly, data warehouses are here to stay, just as most of the legacy applications that they integrate. Due to their increasing complexity, however, they have lost one of their crucial properties: flexibility. As business requirements change, both (legacy) applications and data warehouses must adapt and with growing complexity it becomes increasingly difficult to adapt these systems. For instance, a company may wish to launch a new product; as a result, new reports are needed in order to first analyze the potential of this product and second support new processes of producing and marketing the new product. With the data warehouse growing in data scope and architectural complexity, it becomes harder to collect all necessary information to program these reports as it is not clear whether the required data is even available in the data warehouse and if so, how to best access that data. As a second example, a (legacy) application may have to be adapted because of new regulatory requirements, a common use case in the financial industry. It is not obvious how this change will affect concepts and reports provided by a data warehouse. These examples demonstrate that while the business becomes increasingly dynamic asking for more information and new processes, the IT becomes increasingly slower and the gap is growing.
The purpose of this paper is to show how Credit Suisse has addressed this widening gap in the speed that the business changes and the speed that the IT organization can deliver these changes. We report on the design and implementation of the Credit Suisse meta-data warehouse that has been productive and used by a still growing community of business and IT users within Credit Suisse since 2009. Several releases of the system systematically organized the meta-data and increased its coverage of the organization. Powerful services were built on top of that meta-data to support critical business processes of the bank. Therefore, the meta-data includes concepts used by business users (e.g., Customers, Transactions, etc.) and the concepts used by the IT organization in order to implement these business concepts (e.g., Table, Column, etc.). Furthermore, it contains the relationships between the concepts used by business users and the concepts used to implement them.

There are different ways to build such a meta-data warehouse. The traditional approach is to model the meta-data, structure it using the relational data model, and store and query all meta-data in a standard SQL database system. Unfortunately, this approach does not work well for Credit Suisse because we do not really oversee the full meta-data landscape and all of its relationships we might need. Therefore, we do not know how to create a respective relational data model of the meta-data. Instead, we would like to incrementally build up the meta-data warehouse, thereby adding more and more meta-data as it becomes available and without being constrained by a structured meta-data model. As an alternative, we modeled meta-data as a graph and provided some conventions on how to add meta-data to that graph. Specifically, we have chosen to use RDF [2] to persist and access our meta-data graph because it is standardized and because of the availability of industry-strength tools and systems to manage RDF data. Furthermore, there are interfaces in these tools to convert RDF to UML and other modeling standards that can be used in a model-driven approach. In addition to its flexibility to add and manage new kinds of meta-data, one of the advantages of a graph-based approach to store meta-data is that it provides a uniform way to model meta-data generated and used by business and IT users and the meta-data relationships between them.

This paper reports on the following contributions:
• give concrete examples of meta-data maintained by a global player from the financial industry;
• show how this meta-data can be organized in a flexible way, thereby integrating new kinds of meta-data;
• and describe already implemented use cases that show how meta-data can help to bridge the gap between business and IT users.

Furthermore, we hope that this work will inspire both academics and practitioners who seek novel ideas as well as best practices on managing meta-data and how to make use of it in large organizations.

The remainder of this paper is organized as follows: Section II gives an overview of the IT landscape of a large financial institution like Credit Suisse. Section III describes the graph-based approach to store and manage meta-data used at Credit Suisse. Section IV describes two use cases that show how Credit Suisse currently exploits its meta-data warehouse. Section V reports on the feedback of actual users of the system, working in both business and IT units of Credit Suisse. Section VI summarizes related work. Section VII contains conclusions and possible avenues for future work.

II. CREDIT SUISSE APPLICATIONS

In order to understand the need for comprehensive management of meta-data, this section gives an overview of the meta-data in the IT landscape of Credit Suisse. We believe that the situation is typical for most large companies in the financial and other sectors. Figure 1 gives an overview of some of the most important subject areas of the IT landscape at Credit Suisse. Applications are in the center of attention. Each application serves a particular business or regulatory need: e.g., making payments, managing bank accounts and portfolios, etc. Credit Suisse has several thousand applications and despite the availability of so-called standard applications (e.g. SAP), the number of applications (and their complexity) is still growing, mainly due to the extremely dynamic nature of the financial industry.

Almost every application at Credit Suisse is integrated with a database, where the application’s data structures are defined. The meta-data warehouse keeps track of both the database schema as stored in the database and the schema of information objects used in business process descriptions (i.e. data definitions). An application can have one or more interfaces to other applications as part of a service-oriented architecture. An interface is the physical interaction between two applications. Such interfaces exist between applications and data warehouses, for example applications that feed data to the data warehouse. Information about interfaces is also kept in the meta-data warehouse.

Figure 1 depicts another important meta-data subject area, data flows. Based on the interfaces among numerous applications, data can flow through various paths i.e. from an internal database to an outbound interface and from there to an inbound interface and a second internal database. If a mapping specifies and defines the rules of how data is transformed and flow between two applications, then a group or a chain of mappings define a data flow. Lineage is the process of observing and
determining the distinct states, transformations and flows that data has undergone until it reached a target system.

As an example, Figure 2 shows how Customer data is processed by one of the Credit Suisse data warehouses as part of a private banking application. Figure 2 depicts, from top to bottom, three areas of a data warehouse. The first area, called “DWH Inbound Interface” in Figure 2, collects the data as it is generated by the applications that generate Customer information. This area is often also called staging area in the DWH literature. In this area, there is a specific data model that represents all entities of type Customer. Figure 2 only lists the customer_id field that is used in the staging area to identify customers uniquely.

The second area of the DWH shown in Figure 2 integrates and cleanses data. In this particular example, data from different types of customer applications is integrated: people and organizations. People are referred to as Individuals and organizations are referred to as Institutions by the business and IT users of this integration area. Furthermore, customers are generally referred to as Partners in this area and the data model integrates the different kinds of Customers by using the generalization concept in order to factor out the common properties of Individuals and Institutions. In particular, all Partners are referenced by their partner_id which is an integer. Figure 2 shows how information about Individuals is derived from the Customer information maintained in the staging area. As part of this data flow, a mapping must be defined that transforms the customer_id from the staging area (a string) into a unique partner_id in the DWH integration area. (Institution is derived from other instances of the staging area; that data flow is not shown in Figure 2 for brevity.)

Finally, Figure 2 also shows how Customer information is used in further stages of the DWH such as data marts of the DWH which are used in tools that define reports, support certain operational BI applications, and help make decisions. In this example snippet, a respective interface to a data mart is sketched that refers to all customers as Clients. Such client information is derived from the Partner information in the DWH integration area and compromises information about both Individuals and Institutions.

Going back to Figure 1, Roles are an important concept to manage the application landscape. This subject area models the relationships of users to applications. Of course, roles are used in order to implement authorization rules that specify which user or group of users is allowed to perform which kind of operation in a discretionary access control model. Authorization is the classic, traditional use of roles. At Credit Suisse, however, roles are also used to describe business relationships. For instance, each application has an owner; i.e., a specific user who plays the business owner role and, bluntly spoken, pays the bills and takes the flames of the upper management if something goes wrong. The role business user specifies users who approve a new business process or changes to an existing business process. Other roles include consultant, investment banker, accountant, etc. Some of these business users are internal (i.e., employees of Credit Suisse) and some of these users are external (i.e., contractors) and often users play different roles for the same or different applications. On the IT side, typical roles played by users are administrator or support. These IT roles can be tied to specific applications, but they can also be tied to infrastructure components such as the Oracle databases or security. Again, the meta-data warehouse needs to keep track of all these roles and their responsibilities and rights at both the technical level (e.g., access rights) and the business level (e.g., IT governance).

Figure 1 is a strong simplification and only models a small part of the Credit Suisse meta-data IT landscape on an aggregated level. Some subject matter areas have been omitted as they are not covered by the meta-data warehouse, yet, due to prioritisation by Credit Suisse. This does not imply by no means that additional meta-data is not used for additional use cases in Credit Suisse. For instance, the content of log files; every application and database maintains a log of events which may be subject to inspection by auditors. Moreover, the subject areas also cover meta-data on a physical level such as programming languages and third-party software used to assemble applications. Meta-data on this granularity level is not included in the figure although it is part of the meta-data warehouse. In other words, though not explicitly shown in the figure, a great deal of technical meta-data is actually in use. An increasing number of use cases ascribe growing importance to this type of meta-data. For example data governance use cases: the assignment of owners and consumers of data to meta-data
The meta-data warehouse presents the business and IT with an effective and valuable solution to document their assets and components. Having most kinds of meta-data available in the same application (i.e., the meta-data warehouse) enables IT and business to detect connections between components they might have never seen before by referencing single meta-data repositories only.

In summary, Credit Suisse has a complex and highly dynamic business and technology landscape. Business users are faced with constantly changing information. The application landscape changes continuously, thereby adapting to the information needs of the users. Like most other large organizations, Credit Suisse maintains several data warehouses. Additionally, a meta-data warehouse keeps track of all information items stored in the data warehouses and those applications that feed the data warehouses. Business users who wish to create a new report can query the meta-data warehouse in order to find out whether the required information is stored in a data warehouse with the appropriate freshness, granularity and data quality. They can furthermore understand how a particular report was generated (e.g., from which applications). The information lineage - the capability to observe how information flows through different applications - is particularly useful for auditors who analyze logs in order to assess compliance. Information lineage is critical to understanding how changes to an application or its interface may impact other applications or reports generated from the data warehouses. Moreover, charts provided by data warehouses gain credibility as they can be tracked down by following actual data flows within data warehouses. The increasing transparency of data warehouses is essential for them to be qualified as a strategic asset and premise for major business decisions.

III. Meta-data Management

One approach to manage data would be to construct a relational data model from the diagram shown in Figure 1 following the textbook approach of conceptual data modeling [3]. This way, standard (SQL) database systems could be used to store and query the meta-data efficiently. Clearly, this approach would promise best performance and low operational cost as Credit Suisse already has enough IT experts that know how to administer and program relational database systems. Unfortunately, this approach is too rigid and it requires a major investment in constructing a comprehensive meta-data schema; an investment that Credit Suisse was not willing to make.

The alternative is to store all the meta-data in a (big) labeled graph, i.e. still using a relational database system (Oracle in the case of Credit Suisse) but with extensions that provide features to store and query such a graph. In fact, that approach adds one abstraction layer on top of the relational database system and relies on a generic schema. It provides a great deal of flexibility as nodes and edges can be added and removed in any way. The drawback of this approach is that it might be difficult to answer queries using such a graph if the graph has no structure at all. In order to get the best of both worlds (predefined structure vs. genericity), we propose to use a graph in order to store the meta-data (for its flexibility), yet organize the nodes and edges of the graph in a specific way (for query processing). This section describes this approach and shows how we implemented it using RDF and the Spatial Option of the Oracle database system [4]. (The Spatial Option provides RDF and Semantic Web support.)

A. Meta-data Model

We define meta-data as data, which describes other data, i.e. data about data. This mainly addresses data also referred to as reference data in the business world and data about data structures and data flows in the technical world; including the respective hierarchies as well. Credit Suisse stores four kinds of meta-data in the meta-data warehouse graph that are explained best by using examples. They are modeled as node types (x-axis of Table I) in the form of Instances, Values, Classes, and Properties both for the business world and the technical world:

- **Classes**: For business users, typical classes are Customer, Transaction, or Account. Typical Classes of the technical world are Table, Application, or Role.
- **Properties**: Typical properties are CustomerName or TransactionAmount for the business and RolePrivileges or RoleName for the technical world. Attributes of classes are typical examples of properties.
- **Instances**: Examples of business related instances are particular customers; e.g., John Doe or a specific organi-

<table>
<thead>
<tr>
<th>Classes</th>
<th>Facts</th>
<th>Properties</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>META-DATABASE SCHEMA</td>
<td>Class Nodes</td>
<td>Edges (Class, Property)</td>
<td>Property Nodes</td>
</tr>
<tr>
<td>META-DATA SCHEMA</td>
<td>Edges (Class, Property)</td>
<td>Edges (Property, Property)</td>
<td>Edges (Value, Property)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
<th>Properties</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facts</td>
<td>Edges (Class, Instance)</td>
<td>Edges (Property, Value)</td>
</tr>
<tr>
<td>Properties</td>
<td></td>
<td>Edges (Instance, Instance)</td>
</tr>
<tr>
<td>Values</td>
<td></td>
<td>Edges (Class, Class)</td>
</tr>
</tbody>
</table>

TABLE I
META-DATA WAREHOUSE GRAPH OBJECTS: NODE TYPES ON THE X-AXIS, EDGE CATEGORIES ON THE Y-AXIS AND ACTUAL TYPES OF NODES AND EDGES IN THE TABLE’S CELLS
Instances in the technical world are, e.g., specific database tables.

- **Values:** Examples of values are scalar values (e.g., 100 or USD 100) or strings such as "Zurich" for the business part and strings that represent the names of database tables for the technical world. Unfortunately, many table names at Credit Suisse are quite cryptic such as "TCD100" due to technical restrictions on the length of table names in legacy systems.

Instances, values, classes, and properties form the set of nodes in the Credit Suisse meta-data graph. Again, the Credit Suisse meta-data graph is one big logical graph that represents all meta-data of the business and technical worlds. Relationships between instances, values, classes and properties are stored as edges. Just as the nodes, the edges are organized into three categories, listed along the y-axis in Table I:

- **Facts:** Facts primarily describe the relationships of instances and values. A typical fact is that a specific customer, say, "John Doe" belongs to the **Customer** class. Specifically, facts contain relationships between two instances, or an instance and a value, or an instance and a class, or a value and a property.

- **Meta-data schema:** The meta-data schema contains relationships between properties and classes. For instance, the property **hasFirstName** is an attribute of class **Customer**. In RDF lingua, this is implemented by stating that the domain of **hasFirstName** is class **Customer**. Basically, this part of the graph describes the classes. Again, the meta-data warehouse does not have a fixed **meta-data model** (i.e., only the RDF model needs to be followed, which is very generic); instead, the meta-data schema is flexible and described as part of the database. This way, new classes can easily be added and new properties can be added to classes in a flexible way. The actual definition of classes and their properties is already beyond the scope of RDF and requires the usage of the next level in the Semantic Web Stack: RDF Schema (RDFS) [5].

- **Hierarchies:** This part of the meta-data warehouse describes relationships between classes and properties, mostly sub-class relationships. The purpose of these relationships is to model the terms or concepts (i.e. the classes) that business users may use in their search. For instance, an **Individual** is a **Party** so that a search for **Party** includes looking for **Individuals** (Section II). This area requires concepts beyond RDFS if it comes to more sophisticated definitions of subclasses and properties. For instance, instances of class **Party** with a property **hasFirstName** belong to the class **Individual**. Some properties might be symmetric such as **isRelatedTo**. Such symmetries are not supported by RDF, but they are supported by the Web Ontology Language (OWL) [6].

Figure 3 depicts a snippet of the meta-data warehouse, thereby continuing the **Customer Identification** example from Section II. It shows the data flow with its mapping at the lowest layer, the fact layer. That is, a **Client Information Id** is mapped to a **Partner Id** which in turn is mapped to a **Customer Id**, as described in Section II. **Client Information Id**, **Partner Id**, and **Customer Id** are instances of different classes of the meta-data schema. The **Client Information Id**, for instance, is the column of a source file while the **Customer Id** is a column of a view defined in the data warehouse. As explained later in Section IV, business users are not interested in technical terms such as **Source File Column**; instead, they will be searching for certain applications or interfaces. In order to enable such searches, the hierarchy, the third and highest level of the meta-data warehouse, stores relationships between classes. Figure 3 shows a few of these relationships.

In order to provide an idea on how big and tightly connected the resulting meta-data graph is, we need to consider the time dimension, too. The meta-data warehouse has a full
Fig. 4. Meta-data Warehouse Design with Oracle Semantic Web Technologies

historization mechanism in place, i.e. each meta-data graph is historized completely into a dedicated set of historization tables. There are approximately 130,000 nodes and about 1.2 million edges in every version. The number of versions is following the release cycles of the major Credit Suisse applications, i.e. up to eight versions in one year. But at the same time, the amount of meta-data also increases due to additional meta-data being integrated into the meta-data warehouse. We estimate the current growth rate due to additional sets of meta-data to be about 20 to 30% every year.

B. Meta-data Implementation

Storing meta-data as a graph is a flexible approach: No database schema needs to be devised upfront; instead, the meta-data schemas can be built-up incrementally as more and more meta-data becomes available. Furthermore, instances, facts, meta-data schema, and hierarchies can be stored and queried in a uniform way. Another important advantage for Credit Suisse is that this approach to model and store meta-data is aligned with Web standards such as RDF, RDF Schema and OWL. As a result, tools such as the Spatial Option of Oracle’s database system can directly be applied. There are various ways to use RDF and the Semantic Web stack. We believe that meta-data warehousing as done at Credit Suisse is one of the most compelling application areas for Semantic Web technologies.

Since most of Credit Suisse’s meta-data are available either as XML files or in a format that can easily be converted into XML, the very first step to get meta-data into the meta-data warehouse is to transform it into RDF as shown in Figure 4. This is how those RDF triples that contain the meta-data facts are prepared for the bulk load of all RDF triples into the Oracle database that is the storage facility of the meta-data warehouse. The meta-data hierarchies are designed and maintained in a popular open-source tool called Protégé [7]. They are exported from this tool as an ontology file and inserted as RDF triples into the same staging tables as the meta-data facts. The connection between meta-data facts and meta-data hierarchies is established through the meta-data schema. It is an integral part of both the facts and the hierarchies and it is as well contained in the respective files. Table I defines in its cells the types of nodes and edges that are inserted by these two imports into the integrated logical meta-data warehouse graph.

Physically, all RDF triples are then inserted into RDF model tables by the bulk load. The RDF model tables are similar to traditional relational database tables but with some extensions to support, among others, SPARQL queries [8] and additional indexes for semantic web reasoning. These indexes read all relationships (meta-data schema and hierarchies) and apply them on the basic facts. The resulting derived RDF triples or relationships are included in the indexes. In fact, the indexes add additional edges to the meta-data graph and therefore increase its density. This is particularly useful in cases where some multiple edge paths through the graph could be bypassed by just one additional edge connecting the beginning and the ending node of such a path. Nevertheless, these derived RDF triples do only exist through the indexes. Hence, if a query does not explicitly contain a reference to one of these OWL indexes, then only the meta-data facts are considered for this query. We present example queries taking advantage of OWL indexes in Section IV.

Inside the RDF model tables, we use RDF in the following way. As mentioned in the previous section, instances, values, classes, and properties are represented by nodes in an RDF graph. Furthermore, their relationships are represented as edges in the RDF graph, thereby using predefined RDF, RDF Schema and OWL labels. Concretely the following RDF and RDF Schema labels (i.e., properties) are used:

- rdf:resource: This RDF property is used, for example, to instance relationships (i.e., facts).
- rdf:type: This RDF property is used for instance to class relationships (i.e., facts).
- rdf:property: This RDF property is used for value to property relationships (i.e., facts).
- owl:Class: This OWL label is used to indicate that a node is a class and not just an instance.
- rdf:domain: This RDF property is used for class to property relationships of the meta-data schema.
- rdfs:subClassOf: This RDF Schema property is used for class to class relationships of hierarchies.
- rdfs:subPropertyOf: This RDF Schema property is used for property to property relationships of hierar-
chies.

The only exception to this setup are relationships of type instance to value. They use tags, which are specific to Credit Suisse.

Furthermore, any user-defined labels can be used in order to annotate instance to value and value to value relationships (i.e., facts). Such relationships, for instance, are used to represent synonyms, homonyms, and aggregations. The Credit Suisse meta-data warehouse incorporates meta-data collections from the DBpedia project [9] that extracts subsets of Wikipedia [10]. For instance, links between Wikipedia articles are stored in RDF files and published on the Web. That additional meta-data is used to derive additional edges between synonyms and homonyms in the meta-data graph.

IV. USE CASES

This section gives two examples to demonstrate how the meta-data warehouse is used at Credit Suisse today. The first example shows how business users search for concepts in the meta-data warehouse; search is a critical task whenever a user needs new information that is not readily available in a pre-canned report or she might simply not know where to find it, although she might expect the (meta-)data being already stored somewhere. The second example shows how business users derive the lineage (or provenance) of an information item. This use case, for instance, is important for audits or when a specific application changes.

Now that the basic investment to build a meta-data warehouse has been made, we expect many more use cases to be implemented in the future. For instance, an important use case that is currently under development and that extends the search facility described below is to provide more powerful tools to developers in order to program new reports. In general, there are two steps that need to be taken in order to build a new tool on top of the meta-data warehouse:

- Extend the hierarchy (i.e., class-to-class and property-to-property relationships) in order to make all business terms that are needed by the business users searchable. For business users, classes and properties serve as entry points into the meta-data warehouse so that it is crucial to model all the classes that a business user might be searching for.
- Specify the paths in the RDF graph that are needed to support the use case. Paths in the meta-data warehouse are the analog to the SQL queries used in reports of a regular data warehouse.

The remainder of this section illustrates these two steps for the search and lineage use cases.

A. Search

The search facility is a generic feature of the meta-data warehouse that is used by business and IT users for various purposes. It can, for instance, be used to find all the attributes used in a (bulk service) interface (i.e. a collection of huge files with varying structure) of a particular application. Only few business and IT users actually know all the various kinds of meta-data. For instance, because business users do not know the IT terminology, and the way they could reach out to them by just surfing through the frontend of the meta-data warehouse. Therefore, a powerful search facility is essential to make and keep the meta-data warehouse accessible and usable for the majority of all Credit Suisse users.

A screenshot of the search interface is given in Figure 6. The figure lists some meta-data categories (classes) in order to group search results. A search may include one or several search terms. These search terms represent instances. A typical example of a search term is customer as used in the Oracle
SPARQL query in Listing 1. The background of this question might be a new legal condition or simply to know where customer data is delivered to in order to identify potential data protection and security issues. Users may direct their search to a specific area of the meta-data warehouse by applying filter conditions, i.e. restricting the search in the graph to instances of predefined classes. Since most instances are members of several classes due to multiple inheritance in the meta-data hierarchies, the result is mostly a search for nodes that are instances of an intersection of classes.

As mentioned in the Introduction, Credit Suisse integrates data between applications (as part of an Enterprise Application Integration approach) and in its data warehouses. Within a data warehouse the data is integrated, cleansed, and aggregated in several stages. Specifying the Area allows users to search for meta-data in particular stages of the data integration data pipeline. Finally, search can be carried out at different levels of abstraction; business users typically carry out searches at the conceptual layer whereas IT users may search in the physical layer. Again, the meta-data warehouse keeps track of the schema to which a specific information item belongs to. Going into the details of how this kind of meta-data is modeled is beyond the scope of this paper and highly specific to Credit Suisse so that we omit these details for brevity.

Listing 1. SPARQL Query for a Search for the Term ‘customer’

```
SELECT class, object
FROM TABLE(
  SEM_MATCH(
    {?object rdf:type ?c .
    ?c rdfs:label ?class .
    ?c rdfs:subClassOf dm:Application1_Item .
    ?c rdfs:subClassOf dm:Interface_Item .
    ?object dm:hasName ?term} ,
    SEM_MODELS('DWH_CURR') ,
    SEM_RULEBASES('OWLPRIME') ,
    SEM_ALIASES( SEM_ALIAS('dm', 'http://www.credit-suisse.com/dwh/mdm/data_modeling#') ,
                 SEM_ALIAS('owl', 'http://www.w3.org/2002/07/owl#')) ,
    null )
WHERE regexp_like(term, 'customer', 'i')
GROUP BY class, object
```

Figure 6 shows a screenshot of search results as retrieved by the Oracle SPARQL query in Listing 1. The results or objects as they are called in the query, are grouped by classes that may be of interest to the user. Some of these classes refer to classes that may be of interest to business users (e.g., Application) as they are more generic terms; therefore, they do not require any knowledge of a particular terminology used in a specific schema. Other classes refer to classes that may be of interest to IT people (e.g., Source Column) who know the respective specific (technical) terminology. Based on such a search result, the user navigates to the selected classes (and its actual instances) that she might be interested in. In the screenshot of Figure 6, for example, the user may navigate into Applications and then find application instances that involve the search term (i.e., customer in this particular example).

Again, building this search facility on top of the meta-data warehouse involved two steps. In the first step, the hierarchy was extended in order to represent all concepts that users of the search facility may be using. Unfortunately, this enables business-like filters on the meta-data graph only. But it does not yet support search terms that have not been used in at least one part of the meta-data graph. Hence, as part of this step, meta-data from DBpedia representing synonyms and homonyms might be added to the existing facts to enable semantic resolution beyond simple keyword searching. Second, paths in the meta-data graph need to be determined. Specifically, the search is carried out using the following algorithm:

1) Find all nodes (i.e., classes) in the meta-data hierarchy that are relevant for the search.
2) Find all classes in the meta-data schema that are in the intersection of the hierarchy classes and therefore valid search result types. They are also used later on to group search results as illustrated in Figure 6.
3) Find all instances of those classes (Step 2) as indicated by rdf:type that contain the search term as in the Oracle.

Figure 6. Screenshot of the Meta-data Warehouse Frontend for Search Results

<table>
<thead>
<tr>
<th>Search Result</th>
<th>No. of Results</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Results for &quot;customer&quot;</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application</td>
<td>(21)</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>(22)</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>(33)</td>
<td></td>
</tr>
<tr>
<td>Conceptual Attribute</td>
<td>(21)</td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>(2)</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>Source Application</td>
<td>(21)</td>
<td></td>
</tr>
<tr>
<td>Source Column</td>
<td>(33)</td>
<td></td>
</tr>
<tr>
<td>Source Domain</td>
<td>(2)</td>
<td></td>
</tr>
</tbody>
</table>
SPARQL query in Listing 1. This would be the entries shown if one of the list entries in Figure 6 is expanded. That is, for the search facility rdf:type is the path that drives the search.

This search algorithm is best explained using the example of Figure 5. Figure 5 illustrates the search for customer in the meta-data snippet of Figure 3. In Step 1 and 2 of the algorithm, the search is narrowed down to the Application1_View_Column class as illustrated by the gray rectangles. They indicate that the intersection of all of them is the set of valid meta-data schema classes; in this case this is Application1_View_Column only. In Step 3, the customer_id node is found (possibly among many other instances of the same class that contain the search term). Since there is an instance of Application1_View_Column that matches to the search term, the result set is not empty. Moreover, the customer_id node has inherited its membership in all parent classes of Application1_View_Column such as Attribute and is therefore also part of the group of results for all these classes as indicated in Figure 6.

**B. Lineage**

Figure 7 shows a screenshot for the provenance tool implemented on top of the meta-data warehouse. This tool is useful, for instance, in order to carry out audits; an auditor may want to know which applications (and correspondingly which roles and users) have access to a particular information item (e.g., the balance of a bank account of a user from the USA). Lineage is also critical to manage the evolution of the IT landscape. If an application or interface evolves, it is crucial to understand which other applications and interfaces are affected by this change. Moreover, the actual source of a particular figure in a business report might be crucial to interpret this figure, e.g., it shows results of internal cost accounting or external payments.

Like search, lineage starts with a search for relevant classes of the meta-data schema; in this case, we need to identify those classes that define the target. Unlike the search algorithm, the next step is not to look up immediate instances of the classes of the meta-data schema that were found, but to figure out all instances of these classes that have one to many edges of a particular type (i.e. `isMappedTo`) to other instances that do match with the search term as used in the Oracle SPARQL query in Listing 2.

The user interface used as an entry point for the provenance tool is different to the search frontend. The difference lies in the fact that the provenance tool is much more based on navigation, i.e. its main purpose is to switch from one schema to another through specific links, and does not contain any free text field as the search. In fact, the user can either adjust the scope of the data flow or the granularity level of the information items. These items are either shown on the left as source objects of the data flow or on the right as target objects of the same. Figure 7 shows the results of such a drill-down in the provenance tool that can be applied either on the left (source) side or on the right (target) side. Any combination of left and right hand side is possible until the most detailed level is reached, i.e. data flows from attributes to attributes. Again, users may navigate up and down in order to find more details about the data flow between two schemata (e.g. an application schema and its outbound interface schema) or extend or reduce the scope of the tool from one of the current schemata to the next in the flow (e.g. from an application schema and its
What makes the provenance tool special is that it uses a different algorithm and, specifically, a different path in order to find matching nodes in the meta-data warehouse. Specifically, lineage is implemented using the following algorithm:

1) Find all nodes (i.e., classes) in the meta-data hierarchy that are relevant for the target.
2) Find all classes in the meta-data schema that are in the intersection of the hierarchy classes and therefore valid target types.
3) Find all instances of those classes that are relevant for lineage back to the source node client information id, i.e. that have an outgoing edge of type isMappedTo as in the Oracle SPARQL query in Listing 2. This would be the entries shown on the left in Figure 7.

That is, for the provenance tool isMappedTo is the path that drives the search.

Listing 2. SPARQL Query for Lineage back to Node 'client_information_id'

```
SELECT source_id, target_id, target_name
FROM TABLE (SEM_MATCH(  
    {?source_id dt:isMappedTo ?target_id .  
    ?target_id rdf:type dm:Application1_Item .  
    ?target_id rdf:type dm:Interface_Item .  
    ?target_id dm:hasName ?target_name}  
    SEM_MODELS('DWH_CURR'),  
    SEM_RULEBASES('OWLPRIME'),  
    SEM_ALIASES(  
        SEM_ALIAS('dm', 'http://www.credit-suisse.com/dwh/mdm/data_modeling#'),  
        SEM_ALIAS('dt', 'http://www.credit-suisse.com/dwh/mdm/data_transfer#'),  
        null)  
)  
WHERE source_id = 'http://www.credit-suisse.com/dwh/client_information_id'  
GROUP BY source_id, target_id, target_name
```

This is best illustrated in Figure 8. Again, this figure depicts the lookup of dependent items of client_information_id. Steps 1 and 2 work in the same way as in the search facility (Figure 5). The difference lies in Step 3; here, the path used can be described by the regular expression: (isMappedTo)* rdf:type. Using this expression, there is a match between the client_information_id which is used in the source file of a data warehouse and any instance of Application1_View_Column to be used in a view of the data warehouse. That view in turn feeds for example a report of a business intelligence application generated out of a data mart. In Figure 8, customer_id is such a match for a respective target.

V. LESSONS LEARNED

We collected user feedback systematically from business and IT users through small surveys and interactive presentations to communities of practice. Although the user groups of the meta-data warehouse and their desired usage of the application are still very heterogeneous, there are some general points they agree on:

1) The design and the implementation of a meta-data warehouse based on Semantic Web technology works very well at Credit Suisse. It scales to a reasonable number of graph nodes as mentioned in Section III and there are no known limitations to use the very same approach in a different (non-DWH) area of Credit Suisse or by any other company of a similar size.
2) The initial meta-data scope as shown in Figure 1 is not sufficient, but the extended scope as depicted in Figure 9 seems to satisfy user communities in Credit Suisse and other companies as well.
3) The implementation of the use cases described in Section IV meets the basic requirements of business and IT users but there is still some functionality outstanding to fully leverage its potential.
The remainder of this section describes these lessons and discusses the extensions to the use cases described in this paper.

The meta-data warehouse as an application has been working successfully in the Credit Suisse DWH area for more than two and a half years. Moreover, the approach and the current implementation of the meta-data warehouse are currently rolled-out to additional business and IT areas of Credit Suisse such as master data management that are technically current implementation of the meta-data warehouse are currently rolled-out to additional business and IT areas of Credit Suisse such as master data management that are technically well-known software vendors such as IBM [15] and Infor-

The search use case turned out to be very useful for IT users who know about the basic concepts and design of the data warehouse. Business users who are usually working closely to the interface between business and IT also reported it as being handy and intuitive. But most business users still miss actual support for (pure) business terminology and respective filter facilities. Consequently, the search has to become semantic to really bridge the gap between business and IT, i.e. the search algorithm has to map business to IT terms in the background.

The data lineage use case is currently not specific enough for in-depth business analysis. Overseeing all potential data paths is mostly not sufficient, especially since the number of paths is growing exponentially with every additional data processing step or stage of the data warehouse. Analysts need to know under which conditions, i.e. due to the execution of which rule chain, data might flow into one particular target report. Basically, rule conditions need to be included as filter criteria when navigating the graph. Consequently, the number of potential data paths to follow the graph, based on some particular filters, will stay small even with a significant number of steps and stages.

VI. RELATED WORK

This work is based on the foundations developed by the Semantic Web community: specifically, the RDF [2], RDF Schema [5], and OWL [6] standards. Semantic Web technologies have been applied to address a number of problems in different disciplines. In the life sciences, for instance, RDF has been used to annotate data at the Swiss Bioinformatics Institute [11]. The application of ontologies to enhance search by augmenting queries with synonyms and disentangling homonyms has also been studied in the past [12]. There has also been work on automatically generating ontologies and meta-data graphs; e.g., the mining of synonyms [13]. Again, all the work is applicable to the Credit Suisse meta-data warehouse. What makes Credit Suisse special is that it systematically collects meta-data and tries to apply it to support a variety of different use cases that could not be addressed in the past.

The data lineage use case, sometimes also referred to as data provenance use case, has already been discussed in depth by [14] for a data warehouse environment. However, the usage of RDF, RDFS and OWL for meta-data graphs enable more general queries on the equivalent to a transformation graph as described by Cui and Widom. For instance, data flow analysis can start at any node in the graph and stop at a set of target nodes somewhere in the graph, which share some particular characteristics. The characteristics may be, that they all belong to the same application and one of its database schemas on a conceptual level. Hence, the data lineage use case is applied on a system, which is so generic, that any other kind of meta-data in the meta-data graph can be used for this use case as well. However, our approach does not support the actual data instance level, i.e. we cannot trace a single datum since the number of potential data paths to follow the graph, based on some particular filters, will stay small even with a significant number of steps and stages.

There are established commercial meta-data products by well-known software vendors such as IBM [15] and Infor-
matics [16]. These products basically follow the text-book approach as outlined in section III. However, there are vendors as well, which offer meta-data framework products as a foundation for a meta-data management solution such as Oracle’s meta-data repository [4], which we use for the meta-data warehouse.

In the database community, meta-data management has been studied as part of the Rondo project [17]. The focus of that work is to define operators and their semantics for the transformation of meta-data models. Obviously, that work is highly relevant to our project. At the moment, however, the focus of our work has been on showing how meta-data can be applied to several real-world use cases.

VII. CONCLUSION

This paper described the design and implementation of the Credit Suisse meta-data warehouse that serves two purposes. First, it makes the whole IT infrastructure of Credit Suisse more flexible. The meta-data warehouse supports several IT processes such as writing a new report in a (regular) data warehouse, managing application interfaces and mappings, and finding dependencies between applications if an application evolves. Second, the meta-data warehouse helps to bridge the gap between business users and IT people. For instance, the meta-data warehouse shows how certain business concepts (e.g., Customers) have been implemented. As another example, it provides transparency so that business users can monitor the progress of IT projects that they have initiated and sponsored.

Moreover, it also provides transparency to business and IT users who want to build new business intelligence applications on top of data warehouses. The meta-data warehouse shows information on which data is already integrated and used by various applications. Hence, sharing the knowledge of consistently integrated and cleansed data in an easily accessible way stimulates data reuse and hence enables better and more consistent business insights across the boundaries of business divisions. Anyway, this is a real opportunity for significant cost savings in this area.

There are many ways to build a meta-data warehouse. One way would be to carry out a classical database-schema-first approach. In this approach, a comprehensive meta-data schema would be designed first and then implemented using the classical (relational) database technology. Credit Suisse chose an alternative approach to evolve the meta-data schema together with the meta-data that is stored in the meta-data warehouse. Concretely, Credit Suisse stores both the meta-data schema and the actual meta-data in a single (RDF) graph. This approach provides great flexibility for adding new meta-data and implementing new use cases. Furthermore, this approach makes the initial investment for the design and development of such a meta-data warehouse moderate as there is no need for an agreement on the right meta-data schema at the very beginning.

We believe that the Credit Suisse meta-data warehouse is one of the first mission-critical application scenarios of Semantic Web technologies in the financial industry. This paper described the overall structure of the meta-data warehouse and gave examples of meta-data stored in the meta-data warehouse. Furthermore, this paper described two tools, search and provenance/lineage, that were implemented on top of the meta-data warehouse.

There are a number of avenues for future work. Credit Suisse has only recently gone live with its meta-data warehouse. We need to find ways in order to measure and evaluate the success of this animal in the overall IT landscape. Flexibility is hard to measure and it is in general notoriously difficult to measure the impact of a tool like the meta-data warehouse on key performance indicators such as total cost of ownership and time to market of new and existing applications and the IT infrastructure (e.g., machines). A second important avenue for future research is to revisit the design decision to evolve the meta-data schema over time. Maybe, we will be able to quickly learn the right meta-data schema after only a few years so that it might make sense to move towards more traditional database technology once such a meta-data schema has been defined. Furthermore, it is not clear how disciplined users will use the flexibility that RDF graphs provide. As we know from regular data warehouses, success kills and the complexity keeps on increasing to a point where it becomes unmanageable; the hope is that the meta-data warehouse will help to manage the complexity and evolution of the whole IT infrastructure for many years to come and will not become a burden itself.

REFERENCES