SCORVoc: Vocabulary-based Information Integration and Exchange in Supply Networks

Niklas Petersen
University of Bonn
Fraunhofer IAIS
Bonn, Germany
petersen@cs.uni-bonn.de

Irlán Grangel-González
University of Bonn
Bonn, Germany
grangel@cs.uni-bonn.de

Gökhan Coskun
University of Bonn
Bonn, Germany
Fraunhofer IAIS
coskun@cs.uni-bonn.de

Sören Auer
University of Bonn and
Fraunhofer IAIS
Bonn, Germany
auer@cs.uni-bonn.de

Marvin Frommhold
University of Leipzig
Leipzig, Germany
frommhold@informatik.uni-leipzig.de

Sebastian Tramp
eccenca GmbH
Leipzig, Germany
sebastian.tramp@eccenca.com

Maxime LeFrancois
École nationale supérieure
des mines de Saint-Étienne
Saint-Étienne, France
maxime.lefrancois@emse.fr

***Abstract*—Advanced, highly specialized economies require instant, robust and efficient information flows within its value-added and supply chain networks. Especially also in the context of the recent Industry 4.0, smart manufacturing or cyber-physical systems initiatives more efficient and effective information exchange in sypply networks is of paramount importance. The Supply Chain Operation Reference (SCOR) is a cross-industry approach to lay the groundwork for this goal by defining a conceptual model for supply chain related information. We describe an semantics-based approach for facilitating information flows in supply networks and enabling a round-trip between the definition of metrics and KPIs as well as they automatized execution and propagation. It is centered around the SCORVoc vocabulary which represents the Supply Chain Council's SCOR standard entirely as an RDF vocabulary. In order to operationally use the vocabulary for analyzing, monitoring and optimizing supply chains (in particular for robustness), we present SPARQL queries, which retrieve the respective information from information systems adhering to the vocabulary. We define concrete test scenarios and implement a synthetic benchmark to demonstrate the practicality of SCORVoc.**

I. Introduction

In the past decades, internal enterprise information systems experienced much technical and scientific advancement. However, comparatively little progress was made to improve the exchange of information between enterprises. Until today, most of the communication between enterprises is done via informal channels, such as emails (including file attachments) or telephone calls. Only tier-1 suppliers of major Original Equipment Manufacturers (OEM) are usually fully integrated into the information exchange and corresponding IT-support (e.g. EDI connections) since these are expensive to deploy and maintain. Informal communication is time-consuming, costly and also very inefficient considering that often crucial information is spread among many different people using each their own format or system.

As an example, the production plans of a factory are highly dependent on the incoming supplies, since just-in-time productions aims to keep the stock as low as possible to reduce *dead capital*. Therefore, the instant communication between manufacturer and supplier, for example, in case of supply shortages is critical. Furthermore, information on the reliability of suppliers is a competitive advantage for each business. However often, monitoring the direct suppliers is not enough, since problems deeper in the supply chain (delays, strikes, outages, bankruptcies) can have a negative effect even on reliable suppliers. Therefore, it is of paramount importance to be able to pro-actively identify critical suppliers and potential threats in the entire value-added network.

With the aim to reach this goal, we need a standardized way to represent information of critical importance for the supply network. Commonly, all activities related to the production and logistics are defined as processes. The variety in company size, industry and business models as well as different viewpoints and granularity requirements make it very challenging to define a common representation formalism for these processes.

Driven by the need for such a standard, the *APICS Supply Chain Council*[1] defined a reference model to allow enterprises to describe their business processes in a standardized way. This reference model *Supply Chain Operations Reference Model (SCOR)*[2] now in its contains industry-agnostic definitions for 201 processes and 286 metrics.

By today, SCOR [1], [2], [3], [4] has become a mature reference model in its 11th revision backed up by many global players (including IBM, HP, SAP). Figure 2 gives a high-level overview of the reference model. The main limitation is that SCOR contains primarily only textual definitions, which do not directly allow to implement SCOR compliant IT solutions.

In order to address this limitation, we present in this paper an approach for making the SCOR reference model executable. Our approach comprises the definition of the SCORVoc vocabulary providing an ontological formalization of the terms and concepts defined by SCOR. We argue that using a light-weight

[1] http://www.apics.org/sites/apics-supply-chain-council/about-apics-scc
[2] http://www.apics.org/sites/apics-supply-chain-council/frameworks/scor

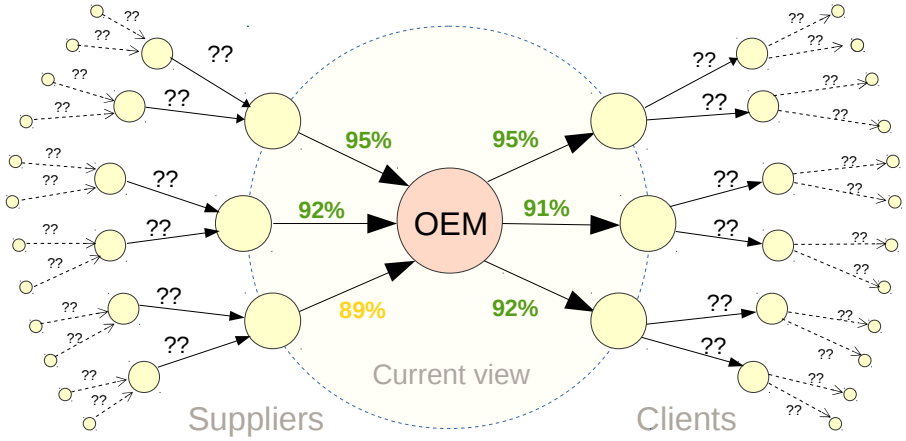


Fig. 1. Supply Chain workflow example: Reliability between Enterprises

RDF vocabulary is a good step towards applying the SCOR reference in real world applications. The fine-grained Linked Data representation formalisms provide a number of benefits for implementing SCOR:

- *Identification.* Word-wide unique identifiers facilitate the data exchange and linking in supply networks,
- *Coherence/Reuse.* Mixing and mashing of vocabularies and schemas enables the reuse and alignment with domain specific formalisms,
- *Granularity.* Integration of representation on different levels of granularity,
- *Execution.* Query execution for automatically aggregating and analysing data,
- *Integration.* Bindings to a number of other technology ecosystems comprising XML (with RDF/XML, JSON (with JSON-LD) or HTML (with RDFa).

As a result, we defined in the SCORVoc vocabulary we defined 211 classes, 257 properties, 28 SPARQL queries [5] and 327 instances with a full documentation together with example use cases. In order to make SCORVoc 'executable' we define queries adhering to the vocabulary for automatically computing crucial KPI's in a supply network, for example, with regard to reliability, responsiveness, agility or cost. In order to demonstrate the practical applicability of our approach, we develop a technique for generating synthetic benchmark data for assessing supply networks. Our evaluation shows, the feasibility of the SCORVoc approach by demonstrating that the amount of data and the query execution is easy to handle for partners in the supply network.

This paper is structured as follows: In Section II we present some use cases which define the requirements for our ontology. We describe our used approach, the design and the usage of SCORVoc in Section III. In Section IV we provide an evaluation of the developed vocabulary and report on related work in Section V. Finally, we discuss future plans and conclude in Section VI.

II. Use Cases

The motivation behind SCOR is to enable enterprises to diagnose and manage their supply chains. Figure 1 illustrates the limited view of an enterprise without any supply chain communication. The goal is to extend the view in order to identify poorly performing links and act upon them. Besides communication, it is necessary that each link is measured equally by each partner. For that purpose, SCOR defined different performance indicators (metrics) including a calculation plan to ensure comparability within the entire supply chain. In total, there are 286 metric grouped together into five categories: *Reliability*, *Responsiveness*, *Agility*, *Costs* and *Assets* (Table I provides a high-level overview on these metrics). The usage of these metrics allow supply chain managers to identify weak and strong links within the supply chain. Therefore, we consider the usage of these metrics as our main motivation for building the SCORVoc vocabulary and the accompanying data integration and assessment approach. The aim is to make the SCOR metrics 'executeable' in the sense, that all information required for computing the metric is made available in a homogeneous way (i.e. as Linked Data) and the metrics itself can be translated into queries operating on this information (i.e. SPARQL queries) and instantly returning the respective KPI. In the sequel, we describe for each category one example use case:

a) Reliability: The metric *Orders delivered in full* (RL 2.1) measures whether orders are received by the customer in the quantities committed.

$$\frac{\sum Orders\ delivered\ in\ full}{\sum Orders\ delivered * 100\%}$$

An order is considered as delivered in full once it contains the correct items (RL 3.33) with the correct quantity (RL 3.35). Thus, this information can be gathered by the corresponding metrics.

b) Responsiveness: The metric *Order Fulfillment Cycle Time* (RS 1.1) measures the average cycle time in days it

Performance indicator	Measures [6]	Result	Example
Reliability	if a task is performed as expected.	Percentage	Order Delivered on-time
Responsiveness	the speed in which tasks are performed.	# of Days	Average days needed to deliver an order
Agility	the ability to respond to external influences.	# of Days	Days needed for an unplanned production increase
Costs	the cost of supply chain processes.	Amount of money	All labour costs required for a specific product
Asset Management	the ability of efficiently utilize assets.	# of Days & Percentage	Inventory days of raw material supply

TABLE I
SCOR PERFORMANCE INDICATOR OVERVIEW

requires to achieve customer orders.

$$\frac{\sum Actual\ Cycle\ Times\ for\ All\ Orders\ Delivered}{\sum Orders\ Delivered * 100\%}$$

Its calculation also depends on multiple other metrics such as the time it takes to procure goods and services (RS 2.1), its production time (RS 2.2), the delivery and the delivery retail time (RS 2.3).

c) Agility: The metric *Upside Supply Chain Flexibility* (AG 2.1) counts the number of days for an unplanned increase (20%) in quantities delivered.

$$\max(Source, Make, Deliver, SReturn, DReturn)$$

By assuming the production can run concurrently (one strategy provided by SCOR), it requires to identify the process (see metrics AG 2.1-5) within the enterprise whose adaption takes the most time.

d) Costs: The metric *Production Cost* (CO 2.004) accounts for all costs involved in the production process.

$$\sum Labor + Automation + Property + Inventory$$

Thus, it depends on metrics which assemble the labor costs (CO 3.014), the automation costs (CO 3.015), the property, plant and equipment costs (CO 3.016) and the the governance, risk, compliance, inventory and overhead costs (CO 3.0017).

e) Assets: The metric *Cast-to-Cash Cycle Time* (AM 1.1) represents the time it takes for an enterprise to earn money on raw material investments.

$$\sum SalesOutstanding + Inventory - PayableOutstanding$$

Thus, it is necessary to summarize the days between a sale is made and the cash is received (AM 2.1) with the days of sales they were in the inventory (AM 2.2). That sums needs to be subtracted with the days between purchasing raw materials and their actual payment (AM 2.3).

III. BUILDING OF SCORVOC

At the core of our approach to make SCOR executable is the creation of a comprehensive RDF vocabulary – SCORVoc – capturing all SCOR related information. Since SCOR is continuously updated, we aimed to provide means for making the vocabulary a living artefact, which can be extended and revised by a community of collaborators. For that purpose, we employed the *VoCol* methodology and support environment [7][3] based on the Git version control system.

Environment Multiple people were involved in the development of SCORVoc. In order to facilitate the collaborative development, we choose a GitHub repository[4] for managing vocabulary source files, documentation, queries as well as example data. The Turtle serialization format [8] was chosen due to its simplicity. GitHub's web interface further provided our domain experts with a very simple way to access the latest SCORvoc version.

Methodology We chose the methodology described by Uschold et al. [9] for building our vocabulary. Thus, first, we defined the purpose and scope. Second, we captured the domain knowledge. Third, we developed the ontology and integrated it with other existing vocabularies. Finally, we evaluated it and documented it properly.

Purpose and Scope The purpose of SCORVoc to provide enterprises with a vocabulary which they can use to express any data related to supply chain management (SCM). The users of the vocabulary are therefore enterprises which would like to profit from the benefits of expressing their supply chains in SCOR. The vocabulary aims at being a light-weight in order to facilitate its usage for future SCOR compliant IT applications.

Capture The capture of the domain of interest (supply chain data management) was achieved in two ways. First, we used the 976-page SCOR reference [6], with its strong terminological definitions as a major source for studying the domain of interest. Second, we had a domain expert with a deep knowledge (a member of the *APICS Supply Chain Council*[5]) which supported us in the entire process. As a result of many interviews with the domain expert, we acquired a more fundamental understanding of the motivation of SCOR, its strengths but also its weaknesses.

1) Design: First, we identified the key concepts for the vocabulary. In SCOR, these are the 201 processes. A process represents any business activity between and within enterprises. For most of them, the reference outlines unambiguous text definitions. Since some of them have a rather long name (e.g. *Identify, Prioritize And Aggregate Supply Chain Requirements*), we decided to keep the short name and attach the long version as a label. To stay coherent, all concept and property names follow the camel case notation.

As proposed in the reference, we created the processes as a hierarchical structure. We defined an abstract super class *Process* with its subclasses *Plan*, *Source*, etc. While certain

[3]http://eis.iai.uni-bonn.de/Projects/VoCol.html

[4]https://github.com/vocol/scor

[5]http://www.apics.org/sites/apics-supply-chain-council

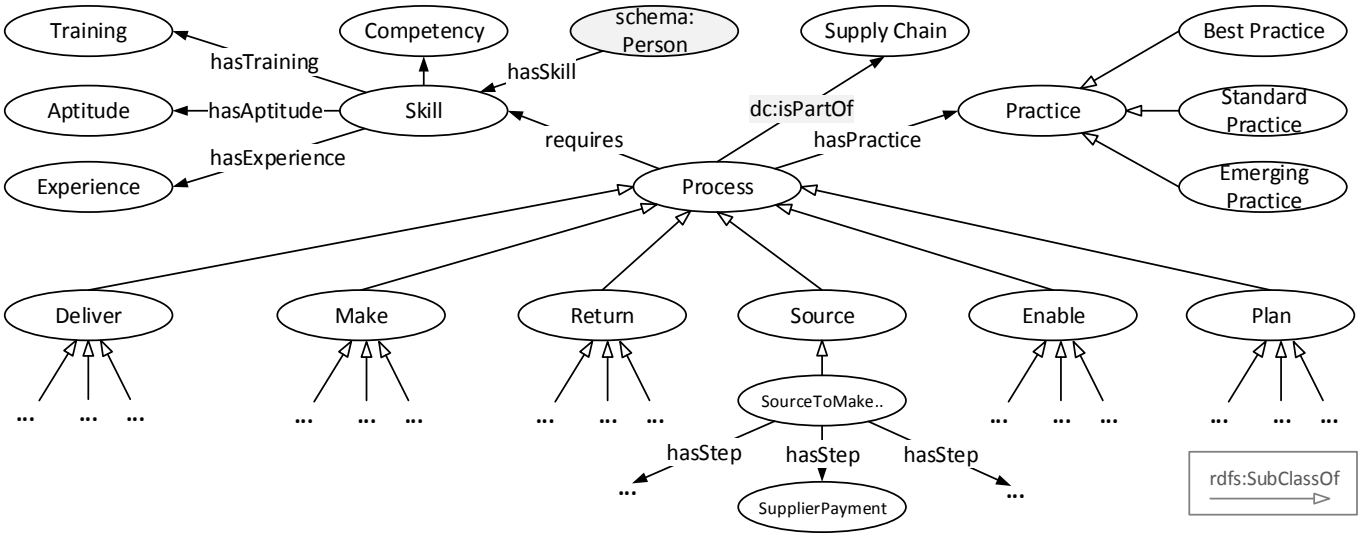


Fig. 3. Overview of the SCORVoc vocabulary (the namespace prefixe `schema` refers to schema.org and `dc` to Dublin Core).

Fig. 2. High-level overview of the Supply Chain Organizations Reference (SCOR).

terms (e.g. *Make*, *Deliver*) do not seem to be self-explanatory, we nevertheless adopted them due to the clear meaning in the domain of SCM. All together, the hierarchy consists of three levels (Scope, Configuration, Step). Each level fulfills a certain purpose:

- Level 1 groups processes together,
- Level 2 comprises events in the real world and therefore are to be instantiated, and
- Level 3 explains in detail how level 2 processes are to be executed (step by step).

Furthermore, the reference defined IDs and clear text definitions for each process and thus was formalized in our vocabulary. Figure 3 shows the general structure of the vocabulary with the processes and others main concepts.

As a next step, *metrics* needed to be addressed. These are defined by SCOR in order to evaluate supply chains on certain aspects such as reliability or responsiveness (cf. the use cases in section II). While metrics are a major term in the reference, we decided to define them as executable queries and properties, instead of defining a concept for each of them as done by previous approaches (cf. related work). SCOR provides for each metric a calculation plan. For instance, the metric *Perfect Order Fulfillment* measures the performance of deliveries. Its calculation plan is defined as:

$$\frac{Total\ Perfect\ Orders}{Total\ Number\ of\ Orders} * 100\%$$

Similar as the aforementioned processes, the SCOR reference organizes metrics into a hierarchical structure (with levels 1-3). Metrics in level 1 and 2 require as an input the results of certain metrics in lower levels. Those metrics where defined as SPARQL queries to allow automatic calculation of the metric value. We consider the level 3 metrics as the *data capture* entry point of our vocabulary which is the rationale for its definition as properties. Their *rdfs:domain* points to their respective processes (given by SCOR) and their range is *xsd:decimal* since they all describe a number between 0-100 (percent values). Finally, the level 1 and 2 queries point to the corresponding property metrics to grab the data necessary for the calculation. Section IV contains multiple examples for SPARQL metric queries.

```
scor:Enable rdfs:subClassOf     scor:Process ;
            rdfs:comment        "Enable describes the ..."
                ;
            rdfs:label          "Enable"@en ,
                                "Permitir"@es ;
            skos:notation       "E" ;
            scor:hasProcessDesc "ProcessType" ;
            skos:altLabel       "sE" .
```

Listing 1. Concept definition example

Moreover, there are a few more relations and classes (yet, less important for the calculation of metrics) within the SCOR model which we added to the vocabulary. As an example, SCOR level 3 processes (also called *steps*) have an order of execution of these processes. Therefore, we used the Ordered List Ontology[6] property *olo:next* to express this relation in our ontology.

SCOR further defines 179 *Practices*. In addition, SCOR defines the section *Person* as a way for managing talent in the

[6]http://smiy.sourceforge.net/olo/spec/orderedlistontology.html

supply chain. This section is divided in the following areas: *Skills*, *Experiences*, *Aptitudes* and *Trainings* and contains 148 elements. We included these definitions as individuals in SCORVoc.

2) Integrating and Alignment with Existing Ontologies: Existing semantics for each concept and the lack of accessibility of prior approaches to formalize SCOR lead us to create many concepts by ourselves. Nevertheless, various concepts are properties are integrated from well-known vocabularies such as *schema.org*, *skos* and *Dublin Core*. We made this decision based on the semantic description of these terms.

3) Documentation: Listing 1 shows an example for the full definition of the concept *Enable*. Enable is a subclass of the abstract concept *scor:Process*. Each concept contains a definition together with further descriptions provided by SCOR. We further added translations for a variety of languages.

Listing 2 shows an example for the full definition of the property *hasMetricRL_33*. Equally as for the documentation of processes, we expressed each property with the definition and the additional information provided by SCOR.

```
scor:hasMetricRL_33  a  owl:DatatypeProperty ;
                rdfs:comment    "Percentage of orders ..." ;
                rdfs:label      "Delivery Item Accuracy"@en,
                                "Exactitud en Entrega de
                                    Items"@es;
                skos:notation "RL.3.33" ;
                rdfs:range      xsd:decimal ;
                rdfs:domain     scor:ItemAccuracyProcesses .
```

Listing 2. Property definition example

IV. Evaluation

We evaluate our approach using a qualitative and quantitative method based on the metrics discussed in Section II. Our qualitative evaluation describes the usage of the SPARQL metrics in a business scenario. In order to do a quantitative evaluation, we developed a SCOR test data generator and measured the execution time of typical queries.

A. Qualitative evaluation

Listing 3 demonstrates a simple example of data expressed using SCORVoc. Since the reference limits itself entirely to processes and performance indicators, we added additional information (using the namespace prefix `yyy`) to make the example more realistic.

The example describes a scenario (*:process_1*) in which goods (i.e. keyboards) are received by an enterprise on a certain date. These goods are forwarded to the warehouse which is the reason for the classification of the process as a *:SourceStockedProduct*. Alternatively, if these goods are directly used in the production or for specialized client orders, the process may have been classified as *:SourceMakeToOrderProduct* or *:SourceEngineerToOrderProduct*. This enables enterprises to distinguish more easily between possible unnecessary orders, which end up as *dead capital* in the stock.

As a next step, all information related to this event is captured. *:hasMetricRL_33* represents the accuracy of the items and *:hasMetricRL_50* the quantitative accuracy. Thus, showing that this order only achieved 90%.

```
#--- General information
:process_1 yyy:isSubjectOf   "Keyboard X"  ;
           yyy:hasTimeStamp "01-01-2015"  ;
           yyy:hasSupplier  :Logitech     ;
           yyy:hasCustomer  :Dell .

#--- SCOR Information
:process_1 a :SourceStockedProduct ;
             :hasMetricRL_33  100  ;
             :hasMetricRL_50  90 .
```

Listing 3. Example of data expressed using SCORvoc

Once the supply chain related information is captured using SCORvoc, the execution of SPARQL query metrics becomes feasible. As described in Section II, this was the driving force in the development of the SCORVoc vocabulary. Listing 4 shows the *Perfect Order Fulfillment* SPARQL metric. The query compares all complete deliveries (achieving 100%) with all deliveries in total by relying on the respective properties. Applied on the previous example, it returns 0% due to the delivery being incomplete.

The knowledge of these metrics is considered to become a major competitive advantage in the enterprise world.

Besides the previous data example, we will present and briefly discuss how the metrics of Section II are realized as SPARQL queries in the following.

The following SPARQL query represents the *Orders Delivered In Full* (metric). Orders are considered to be delivered entirely by SCOR once their items accuracy (RL 33) and their quantitative accuracy (RL 50) are 100% matched.

```
SELECT ((xsd:decimal(?full) / (xsd:decimal(?notFull)) *
     100) as ?result)
WHERE
{   {SELECT ((count(?deliveredInFull)) as ?full)
    WHERE {
        ?deliveredInFull :hasMetricRL\_33 100 .
        ?deliveredInFull :hasMetricRL\_50 100 .
    }}
    {SELECT ((count(?allDeliveries)) as ?notFull)
    WHERE {
        ?allDeliveries a :Process .
    }}
}
```

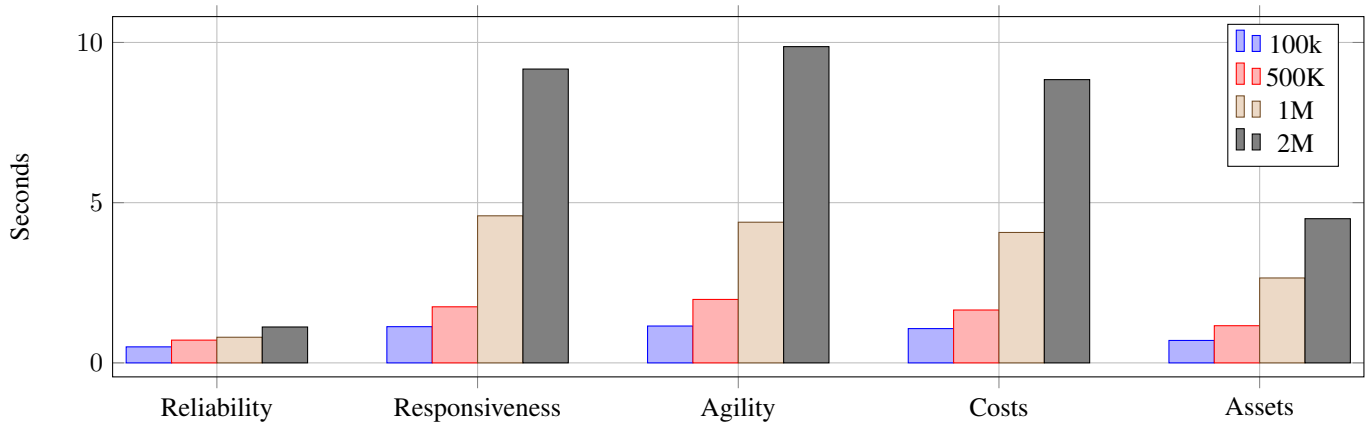
Listing 4. Orders Delivery in Full metric

The following SPARQL query represents the *Order Fulfillment Cycle Time* (metric). The query collects the respective sum (days) of all source (RS 21), make (RS 22), deliver (RS 23) and deliver retail (RS 24) processes and divides them amount of all orders.

```
SELECT ((xsd:decimal(?actualTime)) / (xsd:decimal(?
     allOrders)) as ?result)
WHERE
{
    {SELECT (SUM(xsd:decimal(?value)) as ?actualTime) {
                ?order :hasMetricRS_21
                       |:hasMetricRS_22
                       |:hasMetricRS_23
                       |:hasMetricRS_24  ?value .
    }}
    {SELECT (count(?order) as ?allOrders)
            { ?order a :Process . }}
}
```

Listing 5. Order Fulfillment Cylce Time

The following SPARQL query shows the computation of the *Upside Supply Chain Flexibility* metric. It is necessary to



select all of the flexibility properties (AG1-5) and select the max value. Similar as a team is only as strong as its weakest link, a supply chain is only as agile as its slowest part.

```
SELECT (MAX(xsd:decimal(?flexibility)) AS ?result)
WHERE
{
    ?order :hasMetricAG_1
           |:hasMetricAG_2
           |:hasMetricAG_3
           |:hasMetricAG_4
           |:hasMetricAG_5  ?flexibility .
}
```

Listing 6. Upside Supply Chain Flexibility metric

The following SPARQL query represents the *Production Cost* metric (CO 2.004). This query only relies on the sum of the metric properties for labor (CO 14), Automation (CO 15), Property (CO 16) and inventory (CO17).

```
SELECT (SUM(xsd:decimal(?costs)) AS ?result)
WHERE
{
  ?order :hasMetricCO_14
         |:hasMetricCO_15
         |:hasMetricCO_16
         |:hasMetricCO_17  ?costs .
}
```

Listing 7. Prodction Cost metric

a) Assets: The following SPARQL query represents the *Cast-to-Cash Cycle Time* metric (AM 1.1). The query select thes average time raw materials stays in inventory (AM 2) together with the time the payment is due to by us (AM 1) substracted by that of our customers (AM 3).

```
SELECT (AVG(xsd:decimal(?inventoryDays))
      + AVG(xsd:decimal(?salesOutstanding))
      - AVG(xsd:decimal(?payableOutstanding)) as ?result)
WHERE
{
        ?order :hasMetricAM_1  ?salesOutstanding .
        ?order :hasMetricAM_2  ?inventoryDays .
        ?order :hasMetricAM_3  ?payableOutstanding .
}
```

Listing 8. Cash-to-Cash Cycle Time

B. Quantitative evaluation

In order to demonstrate the feasibility of our approach, we developed a synthetic benchmark data generator. To the best of our knowledge, there are no existing open SCOR datasets. *SCORmark*[7], a dataset assembled by a major consulting firm, is only available for business customers and not open for research. Therefore, we developed a synthetic benchmark data generator, which allows to perform a round-trip between data representation and KPI evaluation. The benchmark allows to assess the performance of an SCORVoc implementation is a systematic and repeatable way. The generator creates data based on a number of parameters – supply chain depth, industry and the number of supply chain partners. The supply chain sets the level from one main OEM enterprise to it suppliers' supplier. The industry generates plausible product lines and enterprise names. The supply chain partners determine the width of the supply chain. A minmum of 2 generates a binary tree to both sides.

Processes	Instances	Related Properties
Source	1.481	28.576
Make	1.785	23.216
Deliver	2.083	22.917
Plan	1.538	18.462

TABLE II
GENERATED DATA OVERVIEW: 100K SCENARIO

Various dataset sizes can be generated in order to assess the scalability as well as the correctness of the metric SPARQL query implementations. We evaluated the metrics for datasets which contain 100k, 500k, 1M and 2M triples. Table II presents an overview on the generated data for the 100k scenario. While the instances represent different types of processes, the related properties are mostly 3-level data type properties which are required by the metrics (such as *:scor:hasMetricRL_50*). The values randomized within a certain range (e.g. >80% for Reliability).

The queries were executed using the *ARQ* SPARQL processor[8]. The machine we used for the experiment contains 8GB of RAM, 256GB SSD and an Intel i7-3537U CPU with 2.00Gz. The generator itself is open-source and available on GitHub[9].

[7]http://www.apics.org/sites/apics-supply-chain-council/benchmarking
[8]https://jena.apache.org/documentation/query/
[9]https://github.com/vocol/scor/generator

	Ye [10]	Fayez [11]	Leukel [12]	Sakka [13]	Zdravkovic [14]	Lu [15]	SCORVoc
Reference Version	7.0	7.0	8.0	9.0	n/a	7.0	11.0
Creation Date	2005	2008	2008	2011	2011	2013	2015
Ontology Availablity	No	No	No	No	No	No	Yes
Completeness	n/a	Assumed	No	n/a	Assumed	No	Yes
Metric Structure	n/a	Hierarchical	Hierarchical	Hierarchical	n/a	Hierarchical	Queries & Properties
Evaluation using data	No	No	No	No	No	No	Yes

TABLE III
EXISTING EFFORTS FOR SCOR VOCABULARIES

V. RELATED WORK

section IV-A summarizes the results of the quantitative evaluation. The *Reliability* and *Assets* queries performed the best in our scenario and were also able to be executed fast with datasets larger 2M triples. The other queries do not scale that well, but still perform with less than 10s query execution time sufficiently good to be employed in real-world settings. Even for much larger supply networks, we deem query execution performance not to be a bottleneck, since queries are executed relatively infrequent and not by thousands of users. Also, since the number of metrics is limited it is possible to optimize query execution even more, but creating specific indexes or applying caching strategies. Overall our evaluation has shown, that the approach of having an executable vocabulary is feasible. The value of our SCOR data generator actually goes far beyond, since it allows SCORVoc compliant software solutions to be systematically assessed and evaluated. We envision, for example, specific supply network visualizations, supply chain robustness assessemtn frameworks, scenario planning tools to be developed based on SCORVoc.

Finally, Figure 4 displays our result: a full view on the entire supply chain.

There exist various works to formalize SCOR into a vocabulary (using RDF and OWL) [11], [12], [13], [14], [15].

The conversion of SCOR model into an ontology is first addressed in [12]. Here, the authors analyze the different conceptualizations levels of the model and convert them into OWL classes. First, the top level which contains the main processes. Second, the Configuration Level which provides a set of process categories for main processes. Finally, the Process Element Level that decomposes the process categories by adding process element definitions and process element information.

In [14], a seminal approach formalizes supply chain operations overcoming the semantic weaknesses of the SCOR model. In this work the SCOR-KOS OWL model is presented, which encodes the main entities and properties for SCOR. In addition, SCOR-Full ontology is constructed as an application ontology and SCOR-Full as a domain ontology. The latter presents the core concepts of Supply Chain embedded in SCOR definitions. This effort is also the basis used by Zdravković et al. [16] to configure the Supply Chain process. They provide a thread model configuring an specific flow of the Supply Chain studied.

The combination of the SCOR ontology and the *ONTO-PDM*[10] ontology is addressed in [15]. The ONTO-PDM ontology is used to represent information regarding product development, which is not covered by SCOR. The goal is to create a Supply Chain ontology framework for networked enterprises interoperability.

Sakka et al. [13] present a SCOR model as a way to align the business processes with strategical objectives for Supply Chains. Concepts like information and input/output are included to face the this alignment. SCOR is modelled using ARIS thus obtaining an SCOR/ARIS model. Then, XLST transforms the output of SCOR/ARIS into a SCOR OWL ontology.

The work conducted by [18] provides an ontology model to support supply chain process modeling and analysis based on SCOR model. In this work only part of the SCOR-KOS model[14] related to the definition of input and output entities in SCOR processes is reused.

Most of these attempts, however, have inherent weaknesses and have not resulted in a publicly available and accessible vocabulary. Contacting the authors to get access to their vocabularies was unsuccessful. As Table III shows these approaches are based on SCOR versions up to 8.0, while the current version is 11.0. Most approaches choose to stay close to the hierarchical structure for processes and metrics given by the SCC textual document. As explained in Section III-1, we are not convinced this close alignment is always necessary and practical. Additionally, the best-practice of reusing and aligning with existing ontologies was not considered in these approaches while SCORVoc provides a basic level of integration with established vocabularies such as *schema.org* and *Dublin Core*.

Another key difference is the rationale to build an 'executable' vocabulary which represents the data efficiently and allows the automatic computation of KPIs. Applying SCORVoc in an industrial setting would generate a huge amount of data since every process represents only a single item. The defined queries with test data shall provide interested parties with use cases and hand-on examples.

VI. CONCLUSIONS AND FUTURE WORK

The use of data-centric approaches in engineering, manufacturing and production are currently widely discussed topics (cf. Industry 4.0, smart manufacturing or cyber-physical

[10]An ontology for Product Data Management interoperability within manufacturing process environment, presented in [17]

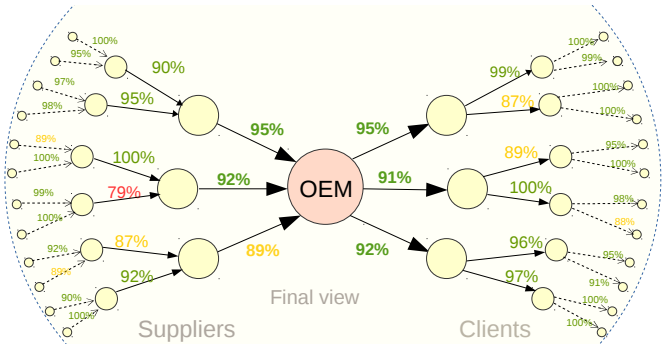


Fig. 4. Final view on the supply chain, where KPIs information is propagated through the network.

systems initiatives). A key issue in engineering, manufacturing and production is efficient and effective supply chain management. In this regard, we introduced a comprehensive approach for facilitating information flows along supply chains centered around the SCORVoc vocabulary. We described its engineering process and provided means for automatically computing typical KPIs. We consider our work together with the formalized SPARQL queries as an important step towards useful SCOR compliant IT applications. SCORVoc is available on GitHub[11] for collaborative further development and at *purl.org*[12].Furthermore, we described comprehensive test scenarios for SCORVoc and implemented a syntetic benchmark.

We see this work as the first step in a larger research and development agenda aiming at providing comprehensive support for information flows accompanying supply chains employing the Linked Data paradigm. As a result of discussions with domain experts, we learned that SCOR has a number limitations. As an example a delivery of 9 out of 10 items is described as a 90% success rate. Indeed, for one company this may be an appropriate measurement, while for another company, the missing part may stop the entire production part. This, among a few other issues, is not yet addressed by SCOR currently. Another step will be the optimization of the queries to proof it usability of them in an industrial setting. Additionally, we plan to add some content ontology engineering patterns[13] as well as the creation of a new Content Pattern for Supply Chain Management as an extension of SCORVoc.

Besides all these technical considerations, we are well aware that in order for any supply chain management solution to work, communication and trust between enterprises is necessary. The more information is shared with the network, the better it can operate.

In conclusion, the complexity in Supply Chain Management in general is thought to be one of the major bottlenecks of the field. SCORVoc aims at reducing this complexity through semantic clarity.

[11]https://github.com/vocol/scor
[12]http://purl.org/eis/vocab/scor
[13]http://ontologydesignpatterns.org/wiki/Category:ProposedContentOP

VII. Acknowledgments

The work presented in this article was partially funded by German Ministry of Education and Research for the project LUCID (http://lucid-project.org/).

References

[1] F. Salazar, M. Caro, and J. Cavazos, "Final review of the application of the scor model: Supply chain for biodiesel castor–colombia case," *Journal of Technology Innovations in Renewable Energy*, vol. 1, no. 1, 2012.

[2] B. Georgise, K.-D. Thoben, and F. Marcus Seifert, "Assessing the existing performance measures, & measurement systems in developing countries: An ethiopian study," *Global Journal of Researches In Engineering*, vol. 13, no. 2, 2013.

[3] F. B. Georgise, K.-d. Thoben, and M. Seifert, "Implementing the scor model best practices for supply chain improvement in developing countries," *International Journal of u-and e-Service, Science and Technology*, vol. 6, no. 4, 2013.

[4] F. Lestari, K. Ismail, A. Hamid, and W. Sutopo, "Designing supply chain analysis tool using scor model (case study in palm oil refinery)," in *IEEE Int. Conf. on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2013.

[5] S. Harris, A. Seaborne, and E. Prudhommeaux, "Sparql 1.1 query language," *W3C Recommendation*, vol. 21, 2013.

[6] S. C. Council, "Supply chain operations reference model," *SCOR, Version*, vol. 11, 2012.

[7] *VoCol: An Agile Methodology and Environment for Collaborative Vocabulary Development*. Zenodo, Feb. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.15023

[8] D. Beckett, T. Berners-Lee, E. Prudhommeaux, and G. Carothers, "Rdf 1.1 turtle–terse rdf triple language. w3c recommendation," *World Wide Web Consortium (Feb 2014), available at http://www. w3. org/TR/turtle*, 2014.

[9] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The knowledge engineering review*, vol. 11, no. 02, 1996.

[10] Y. Ye, D. Yang, Z. Jiang, and L. Tong, "An ontology-based architecture for implementing semantic integration of supply chain management," *International Journal of Computer Integrated Manufacturing*, vol. 21, no. 1, pp. 1–18, 2008.

[11] M. Fayez, L. Rabelo, and M. Mollaghasemi, "Ontologies for supply chain simulation modeling," in *37th conference on Winter simulation*, 2005.

[12] J. Leukel and S. Kirn, "A supply chain management approach to logistics ontologies in information systems," in *Business Information Systems*. Springer, 2008.

[13] O. Sakka, P.-A. Millet, and V. Botta-Genoulaz, "An ontological approach for strategic alignment: a supply chain operations reference case study," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. March 2015, 2011.

[14] M. Zdravković, H. Panetto, M. Trajanović, and A. Aubry, "An approach for formalising the supply chain operations," *Enterprise Information Systems*, vol. 5, no. 4, 2011.

[15] Y. Lu, H. Panetto, Y. Ni, and X. Gu, "Ontology alignment for networked enterprise information system interoperability in supply chain environment," *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 1-2, pp. 140–151, 2013.

[16] M. Zdravković, M. Trajanović, H. Panetto, A. Aubry, and M. Lezoche, "Ontology-based supply chain process configuration," in *34th International Conference on Production Engineering, ICPE 2011*, 2011.

[17] H. Panetto, M. Dassisti, and A. Tursi, "Onto-pdm: Product-driven ontology for product data management interoperability within manufacturing process environment," *Advanced Engineering Informatics*, vol. 26, no. 2, 2012.

[18] T. Grubic and I. S. Fan, "Integrating process and ontology for supply chain modelling," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. March 2015, pp. 847–863, 2011.