A knowledge driven approach towards the validation of externally acquired traceability datasets in supply chain business processes

Monika Solanki and Christopher Brewster

Aston Business School
Aston University, UK
[m.solanki, c.a.brewster]@aston.ac.uk

Abstract. The sharing of near real-time traceability knowledge in supply chains plays a central role in coordinating business operations and is a key driver for their success. However before traceability datasets received from external partners can be integrated with datasets generated internally within an organisation, they need to be validated against information recorded for the physical goods received as well as against bespoke rules defined to ensure uniformity, consistency and completeness within the supply chain. In this paper, we present a knowledge driven framework for the runtime validation of critical constraints on incoming traceability datasets encapuslated as EPCIS event-based linked pedigrees. Our constraints are defined using SPARQL queries and SPIN rules. We present a novel validation architecture based on the integration of Apache Storm framework for real time, distributed computation with popular Semantic Web/Linked data libraries and exemplify our methodology on an abstraction of the pharmaceutical supply chain.

1 Introduction and Motivation

Incorporating novel techniques in their processes that contribute towards eliminating disruptions within global supply chains has been a long standing objective for business organisations and companies. Supply chain visibility (SCV) can be summarised as "Visibility is the ability to know exactly where things are at any point in time, or where they have been, and why"[1]. The goal of SCV is to improve and strengthen the supply chain by making near real-time supply chain knowledge readily available to all stakeholders, including the customer.

However, as useful as this knowledge may be, supply chain data is inherently very sensitive to adhoc integration with third party datasets. For a specific stakeholder, effectiveness of the business workflows and decision support systems utilised within its supply chain operations, that ultimately govern the timely fulfillment of its contractual obligations, is directly dependent on the quality and authenticity of the data received from other partners. Before traceability datasets received from external sources and partners can be incorporated and integrated

[1] http://www.gs1.org/docs/GS1_SupplyChainVisibility_WhitePaper.pdf.

with the supply chain datasets generated internally within an organisation, to be further shared downstream, they need to be validated against information recorded for the physical goods received as well as against bespoke rules, defined to ensure the quality, uniformity, consistency and completeness of datasets exchanged within the supply chain.

In this paper we present a methodology powered by Semantic Web standards and Linked data principles for validating the traceability data sent from one stakeholder to another in the supply chain. Our approach is motivated by four main requirements: (1) The validation should be supply chain domain agnostic, i.e, the constraints must be reusable independently of the goods being tracked. (2) The representation and sharing of traceability data must conform to standards most commonly deployed in supply chains (3) The architecture must be scalable to handle large volumes of streaming traceability data and (4) The constraints must be formalised using widely used Semantic Web standards that are fit-for-purpose. While constraints can be represented using expressive formalisms such as temporal logics, adopting a unified mechanism for representing domain knowledge and constraints eliminates impedance mismatch between the representations, avoids the need for an intermediate mapping language, makes the addition of new constraints easier and simplifies implementation requirements.

Traceability data in supply chains is generated when barcode and RFID readers record traces of products tagged with an EPC (Electronic Product Code), monitoring their movement across the supply chain as specific occurrences of "events". In the proposed framework, description of events is facilitated using EPCIS[2](Electronic Product Code Information Services), a standardised event oriented specifications prescribed by GS1[3] for enabling traceability [4] in supply chains. We exploit two information models: The *EPCIS Event Model* (EEM)[4] based on the EPCIS specification, that enables the sharing and semantic interpretation of event data and *CBVVocab*[5] a companion ontology to EEM for annotating the business context associated with events. In previous work [9] we have shown how EPCIS events can be exploited for the generation of traceability/visibility information that can be shared among supply chain partners as "linked pedigrees". In this paper we show how linked pedigrees received from external partners can be validated against constraints defined using SPARQL queries and SPIN[6] rules. To the best of our knowledge, validating constraints on (real time) supply chain knowledge has so far not been explored both within the Semantic Web and supply chain communities.

The paper is structured as follows: Section 2 presents our motivating scenario from the pharmaceutical supply chain. Section 3 discusses related work. Section 4 highlights the contextual background for the proposed methodology.

[2] `http://www.gs1.org/gsmp/kc/epcglobal/epcis`
[3] `http://www.gs1.org/`
[4] `http://purl.org/eem#`
[5] `http://purl.org/cbv#`
[6] `http://www.w3.org/Submission/spin-overview/`

Section 5 presents the requirements analysis for constraints. Section 6 formalises the constraints using SPARQL and SPIN rules. Section 7 illustrates our execution environment and highlights implementation details. Section 8 discusses the results of our evaluation. Section 9 presents conclusions.

2 Motivating scenario

We outline the scenario of a pharmaceutical supply chain, where trading partners exchange product track and trace data using linked pedigrees. Figure 1 illustrates the flow of data for four of the key partners in the chain.

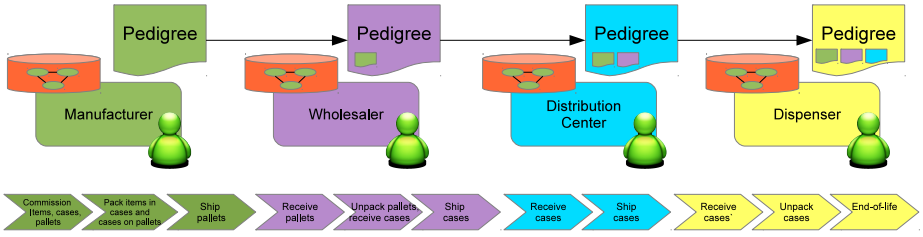

Fig. 1. Trading partners in a pharmaceutical supply chain and the flow of information

The *Manufacturer* commissions[7], i.e., assigns an EPC (Electronic Product Code) to the items, cases and pallets. The items are packed in cases, cases are loaded onto pallets and pallets are shipped. At the Warehouse for the *Wholesaler*, the pallets are received and the cases are unloaded. The cases are then shipped to the various *Distribution centers*. From the Distribution centers the cases are sent to retail *Dispenser* outlets, where they are received and unpacked. Finally, the items are stacked on shelves for dispensing, thereby reaching their end-of-life in the product lifecycle.

EPCIS events are internally recorded for various business steps at each of the trading partner's premises and used for the generation of linked pedigrees. When the pallets with the cases are shipped from the manufacturer's premises to the warehouse, pedigrees encapsulating the set of EPCIS events encoding traceability data are published at an IRI based on a predefined IRI scheme. At the warehouse, when the shipment is received, internal EPCIS events corresponding to the receipt of the shipment are recorded. The IRI of the pedigree sent by the manufacturer is dereferenced to retrieve the pedigree. IRIs of the events corresponding to the transaction (shipping) and consignment (goods) information encapsulated in the pedigree are also dereferenced to retrieve the event specific information for the corresponding business steps. When the warehouse ships the cases to the distribution center, it incorporates the IRI of the manufacturer's

[7] associates the serial number with the physical product

pedigree in its own pedigree definition. As the product moves, pedigrees are generated with receiving pedigrees being dereferenced and incorporated, till the product reaches its end-of-life stage. Note that pedigrees sent by a distributor may include references to the pedigrees sent by more than one warehouse.

3 Related work

The need for standards/frameworks that enable the representation of rules and constraints on the Semantic Web, while complementing the existing ontology representation formalisms such as RDF and OWL has received considerable attention [3] in recent years. Several languages, frameworks and models have been proposed, which mostly have their roots in first order logic (Horn rules), Logic programming, action rules (production rule systems such as Jess, Drools) and Deductive databases. A comprehensive tutorial[8] highlights the state of the art in the fundamentals, applications and standards available for Semantic Web rules. OWL 2 Rules have been explicitly addressed in another tutorial[9].

RuleML[10] is a family of rule languages serialised in XML. RuleML [2] covers a wide spectrum of rules from deliberation, derivation, transformation and reaction rules. It serves as a mechanism to interoperate between certain dialects of other rule specifications such as RIF (Rule Interchange Format), Prolog and N3. SWRL[11] is an expressive rule language that can be used to increase the amount of knowledge encoded in OWL ontologies. SWRL extends the set of OWL axioms to include Horn-like rules. It combines OWL DL with the Unary/Binary Datalog RuleML sublanguages of the RuleML. RIF[12] is a W3C standard designed for interchanging rules between different rule systems. Syntactic mappings that are semantics-preserving can be defined by rule systems from their native languages to RIF dialects and back. The standard RIF dialects are Core, BLD and PRD. OWL-RL[13] is a syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies. OWL 2 RL is ideal for enriching RDF data, especially when the data must be massaged by additional rules.

Provision for the specification of rules have also been made in dedicated Semantic Web frameworks, platforms and triple stores such as Jena[14], OWLIM[15] and Apache stanbol[16]

The use of SPARQL as a constraint/rule language based on its `CONSTRUCT` keyword has long been advocated [7]. SPARQL `CONSTRUCT` is a SPARQL query

[8] `http://silk.semwebcentral.org/iswc2012-rules-tutorial/talk-iswc2012-rules-tutorial.pdf`
[9] `http://semantic-web-book.org/page/KI_2009_Tutorial`
[10] `http://www.ruleml.org/`
[11] `http://www.w3.org/Submission/SWRL/`
[12] `http://www.w3.org/TR/rif-overview/`
[13] `http://www.w3.org/TR/owl2-profiles/#OWL_2_RL`
[14] `http://jena.sourceforge.net/inference/#rules`
[15] `https://confluence.ontotext.com/display/OWLIMv43/OWLIM-SE+Reasoner#OWLIM-SEReasoner-RuleBasedInference`
[16] `http://stanbol.apache.org/docs/trunk/components/rules/`

that returns a graph (set of triples) that is the result of applying the `CONSTRUCT` graph pattern to each match in the `WHERE` clause. Specialised extensions of SPARQL for the rule based processing of events such as EP-SPARQL [1] have also been proposed. SPIN(SPARQL Inferencing notation) is a SPARQL-based rule and constraint language for the Semantic Web. SPIN links class definitions with SPARQL queries using the SPIN Modeling Vocabulary, to capture constraints and rules. SPIN provides mechanisms to capture reusable SPARQL queries using SPIN templates, which are defined as SPARQL queries parameterized with pre-bound variables. SPIN also makes it possible to define custom SPARQL functions.

The use of SPARQL and SPIN to identify data quality problems on the Semantic Web has been proposed in [5]. The authors use SPIN query templates to parameterise the queries with variables. The data quality problems addressed here include missing values, syntax violations and the like. SPIN has also been used for formalising accounting regulations on the Web [6].

Although we explored the various frameworks briefly reviewed above for specifying the type of supply chain constraints we wanted to express, our choice of SPARQL and SPIN was strongly motivated by the fact that a supply chain may have several partners who may well use systems implemented by different vendors. Achieving interoperability between these systems would be crucial in ensuring that the constraints are validated by every partner without which the entire supply chain would be affected. SPARQL is an existing W3C standard with well-formed query semantics across RDF data, has existing widespread use amongst most RDF query engines and graph stores, and provides sufficient expressivity. SPIN allows one to check the validity of an RDF model by using SPARQL and covers a much larger range of potential constraint specifications than other Semantic rules languages. Further, SPIN rules and constraints can be easily shared on the web together with the class definitions they are associated with.

4 Preliminaries

4.1 EPCIS

An Electronic Product Code (EPC)[17] is a universal identifier that gives a unique, serialised identity to a physical object. EPCIS is a ratified EPCglobal[18] standard that provides a set of specifications for the syntactic capture and informal semantic interpretation of EPC based product information. As the EPC tagged object moves through the supply chain, RFID readers record and transmit the tagged data as "events". Given the scenario in Section 2, we are concerned with three types of EPCIS[19] events: *ObjectEvent* represents an event that occurred

[17] `http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf`

[18] `http://www.gs1.org/epcglobal`

[19] Please refer the specification for details.

as a result of some action on one or more entities denoted by EPCs, i.e., commissioning of an object *AggregationEvent* represents an event that happened to one or more EPC-denoted entities that are physically aggregated (constrained to be in the same place at the same time, as when cases are aggregated to a pallet) *TransactionEvent* represents an event in which one or more entities denoted by EPCs become associated or disassociated with one or more identified business transactions, i.e., the shipping of a pallet of goods in accordance to the fulfillment of an order.

4.2 The EEM ontology

EEM is an OWL 2 DL ontology for modelling EPCIS events. EEM conceptualises various primitives of an EPCIS event that need to be asserted for the purposes of traceability in supply chains. A companion standard to EPCIS is the Core Business Vocabulary(CBV) standard. The CBV standard supplements the EPCIS framework by defining vocabularies and identifiers that may populate the EPCIS data model. *CBVVocab* is an OWL ontology that defines entities corresponding to the identifiers in CBV. Development of both the ontologies was informed by a thorough review of the EPCIS and the CBV specifications and extensive discussions with trading partners implementing the specification. The modelling decisions [10] behind the conceptual entities in EEM highlight the EPCIS abstractions included in the ontology. It is worth noting that in previous work [12] we have already defined a mapping between EEM and PROV-O[20], the vocabulary for representing provenance of Web resources. This implies that when a constraint violation is detected, the events in the history can be interrogated using PROV-O for recovering provenance information associated with the events.

The EEM ontology structure and its alignment with various external ontologies is illustrated in Figure 2. The ontology is composed of modules that define various perspectives on EPCIS. The *Temporal* module captures timing properties associated with an EPCIS event. It is aligned with temporal properties in DOLCE+DnS Ultralite (DUL)[21]. Entities defining the EPC, aggregation of EPCs and quantity lists for transformation events are part of the *Product* module. The GoodRelations[22] ontology is exploited here for capturing concepts such as an `Individual` Product or a lot (collection) of items, `SomeItems` of a single type. Information about the business context associated with an EPCIS event is encoded using the entities and relationships defined in the *Business* module. RFID readers and sensors are defined in the *Sensor* module. The definitions here are aligned with the SSN[23] ontology. The *EPCISException* module incorporates the hierarchy of the most commonly observed exceptions [13] occurring in EPCIS governing supply chains.

[20] `http://www.w3.org/ns/prov-o`
[21] `http://ontologydesignpatterns.org/ont/dul/DUL.owl`
[22] `http://purl.org/goodrelations/v1`
[23] `http://purl.oclc.org/NET/ssnx/ssn`

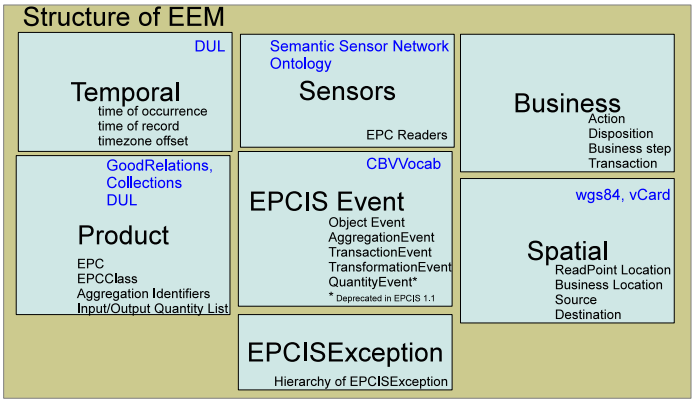

Fig. 2. Structure of EEM and its alignment with external ontologies (noted in blue coloured text)

For further details on EEM and its applications in real world scenarios, the interested reader is referred to [9–12].

4.3 Linked pedigrees

A Pedigree is an (electronic) audit trail that records the chain of custody and ownership of a drug as it moves through the supply chain. Each stakeholder involved in the manufacture or distribution of the drug adds visibility based data about the product at their end, to the pedigree. Recently the concept of "Event-based Pedigree"[24] has been proposed that utilises the EPCIS specification for capturing events in the supply chain and generating pedigrees based on a relevant subset of the captured events. In previous work [9] we introduced the concept of linked pedigrees in the form of a content ontology design pattern, "OntoPedigree". We proposed a decentralised architecture and presented a communication protocol for the exchange of linked pedigrees among supply chain partners. In [11], we extended OntoPedigree to include provenance metadata as illustrated in Figure 3 and proposed an algorithm for the automated generation of linked pedigrees. For the purpose of completeness, we briefly recall the axiomatisation of a linked pedigree in Figure 4.

The definition highlights the mandatory and optional restrictions on the relationships and attributes for every pedigree that is exchanged between stakeholders. Based on these, we define the requirements on the constraints to be validated for the pedigrees.

[24] `http://www.gs1.org/docs/healthcare/Healthcare_Traceability_Pedigree_Background.pdf`

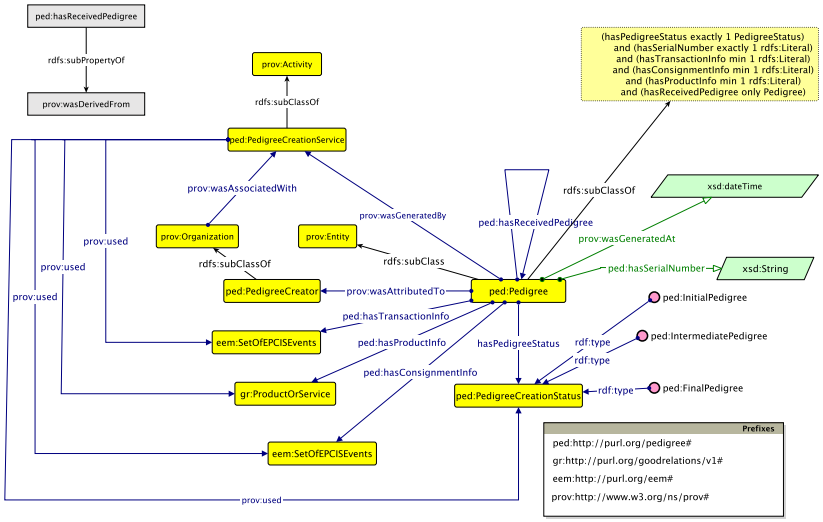

Fig. 3. Graphical Representation of Provenance based OntoPedigree

5 Requirements analysis for pedigree validation

In this section we highlight some of the typical validation requirements on the pedigrees as well as on the traceability data encapsulated in the pedigrees received from upstream partners.

- **Incomplete pedigree**: When a pedigree is received from an intermediate upstream partner it must include deferenceable URIs to the shipping events and the aggregation events in order to verify the upstream CoC/CoO. If the pedigree is received from the initial partner, it must also include dereferenceable URIs to the commissioning events. The Manchester syntax definition of the pedigree illustrated in Figure 4 highlights the entities that are mandatory and must be validated for any pedigree dereferenced by a supply chain partner.
- **Pedigree data has broken chain**: A pedigree received from an intermediate upstream partner must include dereferenceable URIs to other pedigrees from any previous partners who may have been in the line of CoO/CoC for the goods. Validation of included pedigrees must be carried out recursively for the upstream partners until pedigrees from the initial partner in the chain have been validated.
- **Pedigree based, receiving and shipping event correlation**: When a consignment of physical goods is received, the receiving events and the EPCs recorded as part of the receiving events must correlate with the shipping events in the pedigree and the EPCs included in the shipping events.
- **Temporal validity of shipping and receiving events**: For a specific consignment, the shipping events must always precede the corresponding

```
Prefix ped: <http://purl.org/pedigree#>
Prefix prov: <http://www.w3.org/ns/prov-o>

Class: ped:Pedigree
       SubClassOf:
        (hasPedigreeStatus exactly 1 ped:PedigreeStatus)
         and (hasSerialNumber exactly 1 rdfs:Literal)
         and (pedigreeCreationTime exactly 1 xsd:DateTime)
         and (prov:wasAttributedTo exactly 1 ped:PedigreeCreator)
         and (ped:hasConsignmentInfo someValuesFrom eem:SetOfEPCISEvents)
         and (ped:hasTransactionInfo exactly 1 eem:SetOfEPCISEvents)
         and (ped:hasProductInfo min 1),
        (prov:wasGeneratedBy only ped:PedigreeCreationService),
        (ped:hasReceivedPedigree only eem:Pedigree),
        prov:Entity
```

Fig. 4. Manchester syntax serialisation of OntoPedigree

receiving events. Further, this delay needs to be corroborated with the actual time taken for the shipment to be transported from the source to the destination.

– **Missing parent-child aggregation**: In a multi-party supply chain, aggregation and disaggregation of goods may happen at several points. Each of these activities have to be recorded and incorporated within the pedigree. Further goods with EPCs that are part of commissioning events and consequently aggregation events have to be accounted for in every phase of aggregation and disaggregation.

6 Formalising the Pedigree validation rules

In this section we formalise three of the pedigree validation rules identified in Section 5 using a combination of SPARQL queries and SPIN constraints .

6.1 Constraint1: Incomplete pedigree (SPIN constraint)

As per the axiomatisation of a pedigree illustrated in Figure 4, certain attributes and relationships are mandatory. If a pedigree is found to be incomplete, a `PedigreeIncompleteException` needs to be constructed. `PedigreeIncompleteException` is a subclass of `EPCISException` which itself subclasses from `spin:ConstraintViolation`. The textual SPARQL query for validating this constraint can be defined as:

```
#checks for incomplete pedigree
PREFIX ped: <http://purl.org/pedigree#>
PREFIX eem: <http://purl.org/eem#>
PREFIX spin: <http://spinrdf.org/spin#>
```

```
CONSTRUCT
{
 _:b0 a eem:PedigreeIncompleteException;
        spin:violationRoot ?this;
        eem:eventOccurredAt "timeLiteral"xsd:datetime;
        eem:associatedBusinessStep cbv:receiving;
        ....other triples about the exception
        rdfs:label ‘‘Incomplete pedigree exception’’.
}
WHERE
{
     ?this a ped:Pediigree;
           ped:hasPedigreeStatus ?PedigreeStatus;
           ped:hasSerialNumber ?serialNumber;
           ped:pedigreeCreationTime ?pedTime;
           prov:wasAttributedTo  ?pedigreeCreator;
           ped:hasConsignmentInfo ?setOfConsEvents;
           ped:hasTransactionInfo SetOfShipEvents;
           ped:hasProductInfo productInfo;
           prov:wasGeneratedBy ?pedigreeGenerationService;
           ped:hasReceivedPedigree ?pedigree.
    FILTER NOT EXISTS{ ped:hasPedigreeStatus ?PedigreeStatus;
                   ped:hasSerialNumber ?serialNumber;
                   ped:pedigreeCreationTime ?pedTime;
                   prov:wasAttributedTo  ?pedigreeCreator;
                   ped:hasConsignmentInfo ?setOfConsEvents;
                   ped:hasTransactionInfo SetOfShipEvents;
                   ped:hasProductInfo productInfo.}
}
```

The `FILTER NOT EXISTS` clause checks for the mandatory properties, in the absence of which the exception is generated and forwarded to the exception and notification handling modules. A detailed mechanism for representing the triples for the exception itself can be found in [13].

Using the SPIN `spin:constraint` we link[25] the definition of `Pedigree` to the constraint.

```
ped:Pedigree spin:constraint
      [ a sp:Construct;
        sp:templates ([..SPIN generated triples
                        for the construct query...])
      ]
```

[25] Due to space restrictions we do not reproduce the complete definition here.

6.2 Constraint2: Pedigree data has broken chain (SPARQL 1.1 property path)

The property `hasReceivedPedigree` relates a pedigree received by an *intermediate* partner to the pedigree it has received from upstream/downstream partners. We use SPARQL 1.1 property paths to validate the chain of received pedigrees.

```
CONSTRUCT
{
 _:b0 a eem:BrokenPedigreeChainException;
   ..same as the CONSTRUCT above..
   rdfs:label ‘‘Broken pedigree chain exception’’
WHERE
{  ?this a ped:Pedigree;
   ped:hasPedigreeStatus ped:IntermediatePedigree;
   ped:hasReceivedPedigree+ ?pedigree.
 FILTER NOT EXISTS {
        ped:hasPedigreeStatus ped:IntermediatePedigree;
        ped:hasReceivedPedigree+ ?pedigree.}
}
```

6.3 Constraint3: Pedigree based receiving and shipping event correlation

For every pedigree, the set of events corresponding to the `hasTransactionInfo` relationship identify the shipping events. The set of receiving events recorded when the physical goods are received need to have a one-to-one correlation with the shipping events. This implies that the set of EPCs that are part of a receiving event(QueryR) have to be correlated with the set of EPCs in a specific shipping event.

```
#QueryR
SELECT ?epcisR ?epcR
WHERE
{
 ?epcisEventR a eem:EPCISEvent;
              eem:hasBusinessStepType cbv:Receiving;
              eem:associatedWithEPCList ?epcListR.
 ?epcListR    co:element ?epcR.
}
#QueryS
SELECT ?pedigree ?epcisEventS ?epcS
WHERE
{
 ?pedigree a ped:Pedigree;
          ped:hasTransactionInfo ?shippingEvents.
```

```
 ?shippingEvents a eem:SetOfEPCISEvents;
                 co:element ?epcisEventS.
 ?epcisEventS a eem:EPCISEvent;
              eem:hasBusinessStepType cbv:Shipping;
              eem:associatedWithEPCList ?epcListS.
 ?epcListS    co:element ?epcS.
}
```

As noted further(cf. Section 7), the two SPARQL queries are independently executed and the results are combined in a dedicated storm bolt. If the validation fails, the bolt generates the triples for the `PedigreeCorrelationException`.

Validation queries for other constraints can be similarly specified using the various features available in SPIN.

7 Implementing the Validation framework

Figure 5 illustrates the high level architecture that governs our implementation of the validation framework. Pedigrees generated by the upstream/downstream partner are stored in a repository (triple store) and accessed using a queue when a request is made through a Web service. We have used Apache ActiveMQ[26] as our messaging server for the implementation of the queues. A publish/subscribe mechanism is deployed to enable other downstream/upstream partners with relevant authorisation, access control and authentication privileges to subscribe to the pedigrees and dereference them when required. Our validation framework

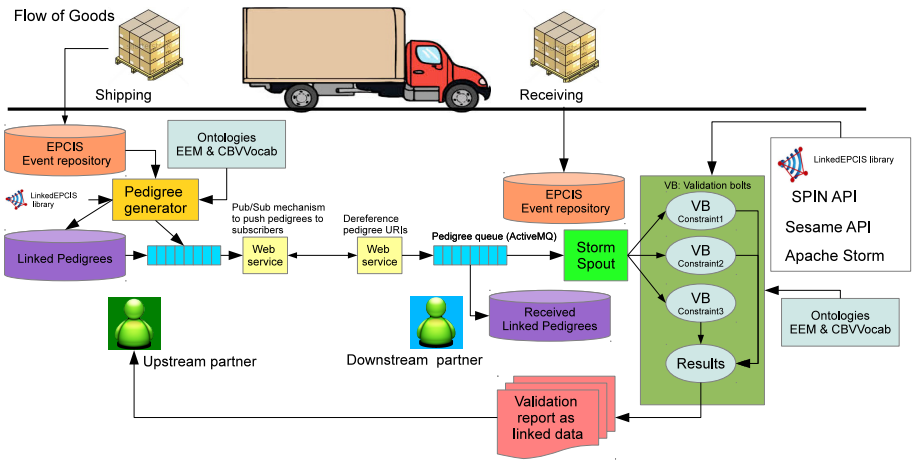

Fig. 5. High level architecture for the validation of pedigree constraints

[26] `http://activemq.apache.org/`

has been implemented using Apache Storm[27], a distributed realtime computation system for processing unbounded streams of data. A novel feature of our validation mechanism is the integration of Storm with external APIs and libraries for carrying out the validations. Specifically we integrate Storm with the SPIN API[28], the OpenRDF Sesame framework[29] and the LinkedEPCIS[30] library developed by us for encoding EPCIS events as linked data. The type hierarchy in LinkedEPCIS is based on the entities defined in the EEM and CBVVocab ontologies.

The validation workflow proceeds as follows: Linked pedigrees are generated using the algorithm and mechanism defined in [11]. When the goods are shipped, the shipping partner schedules the associated pedigrees to be retrieved by the subscribed partners. When the goods are received, the dereferenced pedigrees and the EPCIS receiving events are recorded and sent as streams of data to the validation framework, which has been implemented as an Apache Storm topology. Our Storm spout implementation receives the pedigrees through a queue while each of the bolts is assigned the task of validating a specific constraint. Bolts can execute SPARQL queries and carry out additional computation, e.g., the validation of constraint3 is carried out by first retrieving the results of the two SPARQL queries, performing a comparison between the results of queryR and queryS and generating the exception triples. Results from the validation are sent to the subscribing result bolt which is responsible for generating the aggregated validation results as linked data.

8 Evaluation

Our evaluation for the pharmaceutical scenario outlined in Section 2 focuses on the time taken for the validation of the various types of constraints identified in Section 6.

In order to estimate the volume and velocity of events generated in pharmaceutical event streams, we obtained a large representative sample of EPCIS event data from a well known pharmaceutical company[31], referred to grey literature and interviewed people closely involved in the pharmaceutical sector and EPCIS experts. We referred to a survey [8] that studied the cost benefit analysis of introducing EPCIS event-based pedigrees in the pharmaceutical supply chain. As per the survey, the average volume (number) of pallets, cases and items per month being shipped out of a typical manufacturing unit is 290, 5800 and 580,000 respectively. Interviews with experts corroborated the facts, however they also stated that for some large scale units, the number of items shipped could be as high as 100,000 per day.

[27] `https://storm.incubator.apache.org/`
[28] `http://topbraid.org/spin/api/`
[29] `http://www.openrdf.org/`
[30] `https://github.com/nimonika/LinkedEPCIS`
[31] Named withheld due to data confidentiality issues.

Assuming an average rate of production as 6 days per week and 10 hours per day, and using the sample data, we ran a simulation that replicated the volume and velocity of event generation. We generated the commissioning events based on the number of items ranging from 24,000 to 102,000 per day or approximately 40 to 170 per minute. As the number of items packed per case and the number of cases loaded per pallet could vary across manufacturing units, we generated aggregation and shipping events, considering aggregated items ranging from 100 to 500 (increments of 100) per case and number of cases per pallet ranging from 20 to 100 (increments of 20). We experimented with tumbling window sizes of 3, 5, 7 and 10 hours respectively and generated the linked pedigrees. Overall we ran a total of 400 combinations of commissioning, aggregation and shipping events to generate the pedigrees. For more information on evaluation of the algorithm for the generation of the pedigrees themselves the interested reader is referred to [11]. The event dataset dumps have been made available[32].

Next, for each of the 400 runs, we generated the receiving events using the same procedure as above. For every run, we observed the time taken by the respective storm bolts to validate each of the constraints formalised in 6. The evaluations were made on Mac OSX 10.9.2, 1.7GHz Intel core i5, 4GB 1333 MHz DDR3. The results from the evaluation are illustrated in Figure 6

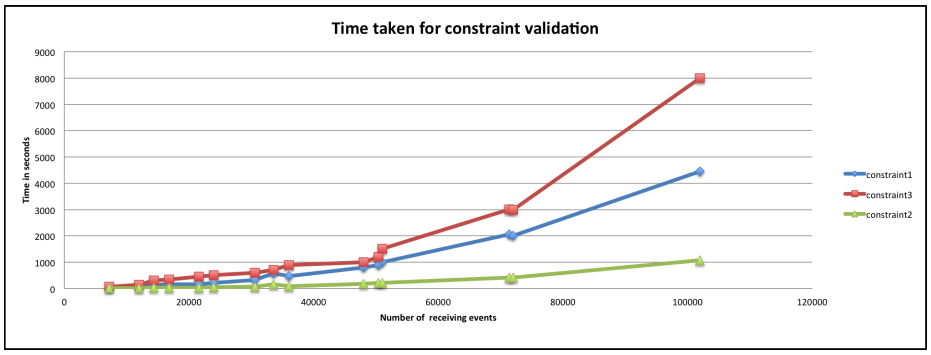

Fig. 6. Time taken for constraint evaluation for varying number of receiving events

As observed, constraint(1) and constraint(3) were computationally more time consuming. The time taken to validate the constraints increased linearly with the increase in the number of receiving events. Further, a major chunk of the validation of constraint(3) was done as part of the processing in the bolt after the query results were obtained, whereas validation of constraint(1) was only a result of the query execution. On the other hand, validation of constraint(2) took comparatively less time and did not vary significantly with the increase in the number of events.

[32] `http://fispace.aston.ac.uk/pharma/eventDatasets`

The differences in the time taken for the validation for each of the constraints, resulted in an overall increase in the validation time. The results show that in order to optimise the performance of the validation framework, a better approach would be to generate each pedigree with low-medium number of events. This would harmonise the time taken for the validation of the constraints and result in increased efficiency of the overall validation framework.

9 Conclusions

The sharing of traceability knowledge between partners greatly enhances their ability to track parts, components or products in transit from the manufacturer to their final destination. In the healthcare sector, visibility of datasets that encapsulate track and trace information is especially important in addressing the problems of drug counterfeiting.

In this paper we have shown how rule based frameworks such as SPIN, driven by Semantic Web standards and linked data libraries can be integrated with distributed realtime computation systems such as Apache Storm to process real time streams of supply chain data. We exploit this knowledge for the validation of constraints that are defined to ensure the quality, uniformity, consistency and completeness of datasets exchanged between supply chain partners. Our constraints are representative of the most commonly occurring scenarios in supply chain and it is worth noting that while we have chosen the healthcare sector as a case study, our approach is domain independent and can be widely applied to most scenarios of traceability.

There are several issues such as trust relationship between actors, access control mechanisms and performance optimisation of distributed storm topologies that need to be considered in supply chains, especially when commercially sensitive data is being shared among partners. In this paper we have abstracted from those issues. Our aim was to show the relevance of validating streaming supply chain event data to the problem of real time tracking and tracking in supply chains.

Much work still needs to be done, especially in making the visualisation of the validation reports intuitive to the partners. We are currently building a Linked pedigrees dashboard that would enable the visualisation of various aspects of linked pedigrees including the violation of constraints and the potential corrective actions taken.

Acknowledgements

The research described in this paper has been partially supported by the EU FP7 FI PPP projects, SmartAgriFood (`http://smartagrifood.eu`) and FISpace `http://www.fispace.eu/`

References

1. Anicic, Darko et al. EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11. ACM, 2011.
2. H. Boley, A. Paschke, and O. Shafiq. Ruleml 1.0: The overarching specification of web rules. In *Semantic Web Rules - International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings*, pages 162–178, 2010.
3. T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres. Rules and ontologies for the semantic web. In Baroglio, Cristina et al., editor, *Reasoning Web*. Springer-Verlag, 2008.
4. K. Främling, S. Parmar, V. Hinkka, J. Tätilä, and D. Rodgers. Assessment of epcis standard for interoperable tracking in the supply chain. In T. Borangiu, A. Thomas, and D. Trentesaux, editors, *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, volume 472 of *Studies in Computational Intelligence*, pages 119–134. Springer Berlin Heidelberg, 2013.
5. C. Fürber and M. Hepp. Using sparql and spin for data quality management on the semantic web. In *Business Information Systems, 13th International Conference, BIS 2010, Berlin, Germany, May 3-5, 2010. Proceedings*, Lecture Notes in Business Information Processing. Springer, 2010.
6. S. O'Riain, J. McCrae, P. Cimiano, and D. Spohr. Using spin to formalise accounting regulations on the semantic web. In *Proceedings of the First International Workshop on Finance and Economics on the Semantic Web (FEOSW 2012)*. CEUR Proceedings, 2012.
7. A. Polleres. From sparql to rules (and back). In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07. ACM, 2007.
8. A. Sinha. Systems engineering perspective of e-pedigree system. Master's thesis, Systems Design and Management (SDM), MIT, 2009.
9. M. Solanki and C. Brewster. Consuming Linked data in Supply Chains: Enabling data visibility via Linked Pedigrees. In *Fourth International Workshop on Consuming Linked Data (COLD2013) at ISWC*, volume Vol-1034. CEUR-WS.org proceedings, 2013.
10. M. Solanki and C. Brewster. Representing Supply Chain Events on the Web of Data. In *Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE) at ISWC*. CEUR-WS.org proceedings, 2013.
11. M. Solanki and C. Brewster. EPCIS event based traceability in pharmaceutical supply chains via automated generation of linked pedigrees. In Peter Mika et al., editor, *Proceedings of the 13th International Semantic Web Conference (ISWC)*. Springer-Verlag, 2014.
12. M. Solanki and C. Brewster. Modelling and Linking transformations in EPCIS governing supply chain business processes. In Hepp, Martin; Hoffner, Yigal (Eds.), editor, *Proceedings of the 15th International Conference on Electronic Commerce and Web Technologies (EC-Web 2014)*. Springer LNBIP, 2014.
13. M. Solanki and C. Brewster. Monitoring EPCIS Exceptions in linked traceability streams across supply chain business processes. In *Proceedings of the 10th International Conference on Semantic Systems(SEMANTiCS)*. ACM-ICPS, 2014.