Detecting EPCIS exceptions in linked traceability streams across supply chain business processes

Monika Solanki
Aston Business School
Aston University
m.solanki@aston.ac.uk

Christopher Brewster
Aston Business School
Aston University
c.a.brewster@aston.ac.uk

ABSTRACT

The EPCIS specification provides an event oriented mechanism to record product movement information across stakeholders in supply chain business processes. Besides enabling the sharing of event-based traceability datasets, track and trace implementations must also be equipped with the capabilities to validate integrity constraints and detect runtime exceptions without compromising the time-to-deliver schedule of the shipping and receiving parties. In this paper we present a methodology for detecting exceptions arising during the processing of EPCIS event datasets. We propose an extension to the EEM ontology for modelling EPCIS exceptions and show how runtime exceptions can be detected and reported. We exemplify and evaluate our approach on an abstraction of pharmaceutical supply chains.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Semantic Web, Linked data, traceability, EPCIS, exceptions, pharmaceuticals, Stream processing

1. INTRODUCTION

The notion of traceability/visibility in supply chains 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]. Barcodes and more recently RFID tags have provided initial solutions to this challenge by recording the traces of product movement as specific occurrences of "events".

The Electronic Product Code Information Services (EPCIS)[2] is an event oriented specifications prescribed by GS1 for enabling traceability in supply chains. The data associated with the business context of scanning a barcode or reading an RFID tag is encapsulated within the abstraction of an "EPCIS event". In previous work [19] 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 build on that approach and propose a methodology that monitors the incoming event stream and the linked pedigrees to detect and report commonly occurring exceptions and violations of critical supply chain constraints within and across supply chain partners. Our streams comprise of EPCIS events annotated using RDF/OWL vocabularies. Each event is modelled as a named graph [5] and annotated with provenance information to facilitate the analysis of the exceptions detected. An exception itself is represented as a special event that can be detected, annotated and reported.

Our exemplifying scenario in the paper, is an abstraction of the pharmaceutical supply chain. In the fall of 2013, the U.S. House of Representatives and Senate passed the Drug Quality and Security Act (DQSA)[3]. The track-and-trace provisions, themselves known as *The Drug Supply Chain Security Act* (DSCSA)[4] within the DQSA outlines critical steps to build an electronic, interoperable system to identify and trace certain prescription drugs as they are distributed in the United States. In readiness for its implementation in the healthcare sector from 2015 onwards, the GS1 Healthcare US Secure Supply Chain Task Force has developed guidelines[5] to identify and serialise pharmaceutical products, in order to trace their movement in U.S. pharmaceutical supply chains. The guidelines are based around the implementation of EPCIS as a standard for event oriented, pedigree (cf. Section 3) track and trace. In accordance to these guidelines, the approach proposed in this paper utilises EEM (EPCIS Event Model)[6] - an OWL DL ontology for EPCIS, as the specification for encoding the event data streams and linked pedigrees for exchanging the event-based traceability information. To the best of our knowledge, runtime monitoring of EPCIS event streams annotated with semantics has so far not been explored in both the Semantic Web and supply chain communities.

[1] `http://www.gs1.org/docs/GS1_SupplyChainVisibility_WhitePaper.pdf.`
[2] `http://www.gs1.org/gsmp/kc/epcglobal/epcis`

[3] `http://www.gpo.gov/fdsys/pkg/BILLS-113hr3204enr/pdf/BILLS-113hr3204enr.pdf`
[4] `http://www.fda.gov/Drugs/DrugSafety/DrugIntegrityandSupplyChainSecurity/DrugSupplyChainSecurityAct/`
[5] `www.gs1us.org/RxGuideline`
[6] `http://purl.org/eem#`

The paper is structured as follows: Section 2 presents our motivating scenario from pharmaceutical supply chains. Section 3 discusses background and related work. It highlights some of the exceptions that can arise in EPCIS governing supply chains and presents an overview of the EEM ontology for describing EPCIS events. Section 4 presents the EPCIS exceptions module as an extension to EEM. Section 5 exemplifies a few SPARQL queries we use for detecting the exceptions. Section 6 illustrates our methodology, execution environment and highlights implementation details. Section 8 presents conclusions.

2. MOTIVATING SCENARIO

We outline the scenario of an abstraction of the 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.

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*, 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, the IRI of the pedigree is dereferenced to retrieve the manufacturer's pedigree. When the warehouse ships the cases to the distribution center, it incorporates the IRI of the manufacturer's 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.

In the scenario described above and as illustrated in Figure 2, runtime exceptions can occur both within a trading partner's premise as well as when a pedigree sent by an upstream trading partner is received, dereferenced and validated by a downstream partner. Exceptions involve certain critical supply chain constraints that could be violated. The runtime traceability event data for pedigrees that is being recorded therefore needs to be validated using business specific rules defined for the transaction. If an exception is detected internally, prompt corrective measures need to be taken in order to avoid significant shipment delays further in supply chains governed by forward logistics. If an exception is detected at a downstream partner's end, it needs to be recorded and reported to the upstream partner. In the following section, we highlight some of the exceptions that are commonly observed in supply chain processes.

[7]associates the serial number with the physical product

3. PRELIMINARIES

3.1 Background

Supply chain track and trace involve a large number of complex operations that may generate several kinds of exceptions related to the data that is included as part of the pedigree. In this context, an exception is an *event* that happens in the supply chain that interferes with or is not part of the expected flow of goods from manufacturer to wholesaler/distributor to pharmacy/clinic/hospital[8].

The GS1 Healthcare US Secure Supply Chain Task Force has identified and documented[9] several of these exceptions. Typical examples include: (e1) Pedigree serial number discrepancy, (e2) product inference problem - the inability to infer about products contained in an outer container without disaggregation using pedigree information (e3) quantity inference problem - the inability to derive the total quantity of items packed in an outer container without disaggregation using pedigree information (e4) missing or incorrect containment hierarchy between items and their containers. This is a critical exception that usually arises in pharmaceutical supply chains when counterfeit products are introduced, for e.g., by replicating the unique identifiers of legally commissioned products and using those for the counterfeit products or by replacing genuine products with counterfeits during the aggregation (packing) phase. (e5) incomplete pedigree data and (e6) pedigree data with broken chains, i.e., missing intermediate stakeholder pedigree information.

Besides these, examples of exceptions have also been highlighted in [1]. These include: (e7) exceptions that may arise due to a lack of velocity consistency, measured in terms of the discrepancy between the timings recorded for the shipping and receiving of a product against the time taken to transport the goods, (e8) lack of dwell-time consistency, specifically important in the case of perishable goods supply chains where goods need to be transported within specific thresholds, (e9) lifecycle consistency, which ensures the correctness of the sequence of business steps recorded as products move between supply chain partners and (e10) pairwise shipping and receiving confirmation, which ensures the traceability of every product shipped and received.

An Electronic Product Code (EPC)[10] is a universal identifier that gives a unique, serialised identity to a physical object. EPCIS is a ratified EPCglobal[11] 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". In this paper, we are particularly concerned with three types of events:

- *ObjectEvent* represents an event that occurred as a result of some action on one or more entities denoted by EPCs. Object events are typically recorded when an item/items is assigned an EPC. The business step associated with the event is "commissioning". e.g., *"This list of objects was commissioned in warehouse #12 at*

[8]http://www.healthcarepackaging.com/playbooks/pharmaceutical-serialization-playbook
[9]http://www.gs1us.org/RxGuideline
[10]http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf
[11]http://www.gs1.org/epcglobal

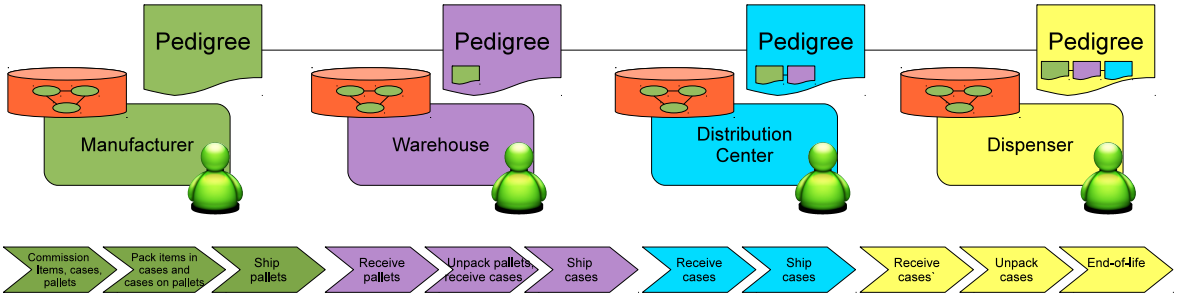

Figure 1: Trading partners in a pharmaceutical supply chain and the flow of information

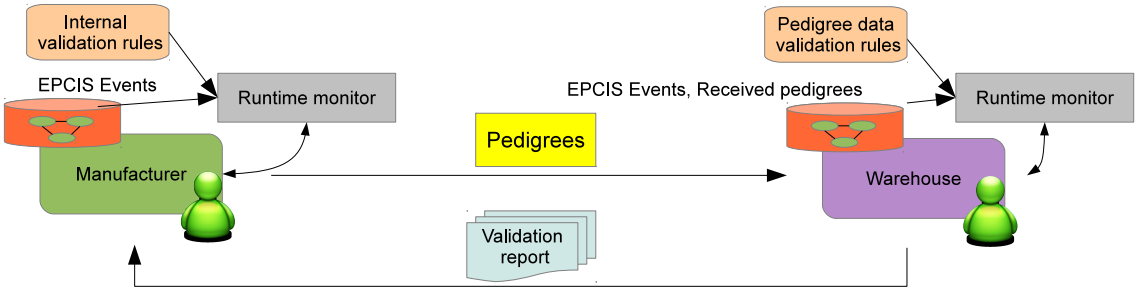

Figure 2: Runtime monitoring in supply chains

12:01AM".

- *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), e.g., *"This list of objects was just Palletized with this Pallet ID at Palletizer #27 at 12:32PM"*. The business step associated with the event is "packing".

- *TransactionEvent* represents an event in which one or more entities denoted by EPCs become associated or disassociated with one or more identified business transactions, e.g., *"Order #123 was fulfilled with objects x, y and z"* .

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"[12] have 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 [19] we introduced the concept of linked pedigrees in the form of a content ontology design pattern, "OntoPedigree", proposed a decentralised architecture and presented a communication protocol for the exchange of linked pedigrees among supply chain partners.

3.2 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*[13] is an OWL DL ontology that defines entities corresponding to the identifiers in CBV. The 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 [20] behind the conceptual entities in EEM highlight the EPCIS abstractions included in the ontology. It is worth noting that in previous work [21]we have already defined a mapping[14] between EEM and PROV-O[15], the vocabulary for representing provenance of Web resources. This implies that when an exception is detected, the event causing and events preceding the exception can be interrogated by the exception handling mechanism 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 3. 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)[16]. Entities defining the EPC, aggregation of EPCs and quantity lists for transformation events are part of the *Product* module. The GoodRelations[17] ontology is exploited here for capturing concepts such as an `Individual` Product or a lot (collection) of items, `SomeItems` of a

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

[13] `http://purl.org/cbv#`

[14] `http://fispace.aston.ac.uk/ontologies/eem_prov`

[15] `http://www.w3.org/ns/prov-o`

[16] `http://ontologydesignpatterns.org/ont/dul/DUL.owl`

[17] `http://purl.org/goodrelations/v1`

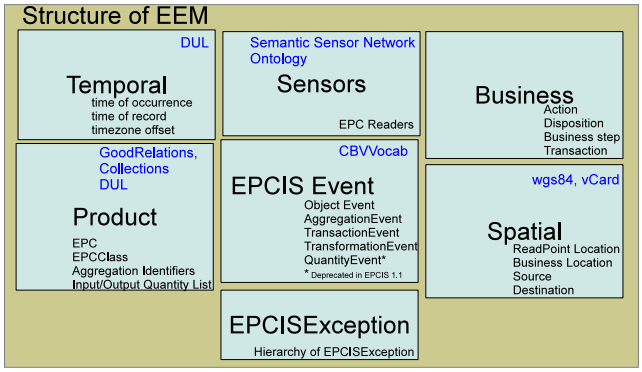

Figure 3: Structure of EEM and its alignment with external ontologies (shown top aligned)

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[18] ontology. For further details on EEM and its applications in real world scenarios, the interested reader is referred to [19–21].

3.3 Related work

Ontologies for exceptions handling in business processes have been discussed in [10]. The taxonomy of exceptions used in this work have been drawn from the MIT Process Handbook[19]. Rules for the detection of exceptions have been specified as situated courteous logic programs. However the approach presented in the paper is very generic and uses DAML+OIL[20]. Our approach on the other hand focuses on the modelling of exceptions in OWL 2 and addresses a very domain specific problem in supply chains.

Exception types for supply chains have also been defined in [11], however they do not cover all the possible exception types for EPCIS and are not available as an ontology to reuse and exploit. An taxonomy for exceptions in business processes have been defined in [8] has been defined in, however they are generic, whereas the taxonomy we process is aligned with EPCIS governing business processes. A multiagent based approach to exception handling in supply chains has been proposed in [15]. The use of ontologies has been proposed but references to it has been provided. The approach is directed towards preemptive exception handling

Monitoring of event-based streams is closely related to the domains of Complex Event Processing (CEP) [14] and data stream management [9]. Several approaches [22, 23] have been proposed that utilise CEP for RFID events. In [22], an event composition language, declarative rules and a graph based event processing model are presented. In [23] the authors synthesize behavioural profiles of business processes for query optimisation, based on external process models that define potential sequences of events. Formal modelling of RFID events and roles for pharmaceutical supply chains has been proposed in [17]. In [4] the authors proposed an extension of an SQL based stream query language with temporal operators for processing RFID data streams. While complex frameworks for event management are available [3], few address the need for representing exceptions explicitly as events.

Within the Semantic Web Community [7], Several frameworks for Complex event processing and querying over RDF streams such as C-SPARQL [6], CQELS [12], EP-SPARQL [3] and INSTANS [16] have been proposed. A crucial difference between the stream processing data model used in the existing approaches and our model is that while most approaches assume streams to comprise of a sequence of time-stamped RDF triples, our streams are sequences of time-stamped RDF graphs. Further, C-SPARQL and CQELS extend SPARQL with sliding and tumbling window operators, and constructs for specifying input and outputs streams. For our requirements of detecting exceptions, window operators would not serve the purpose, as we need to check every event that is generated or that arrives encapsulated in a linked pedigree. EP-SPARQL on the other hand closely fits our purpose. It extends SPARQL with event processing capabilities and operators from Allen's interval algebra [2] such as sequence (`SEQ`) for combining simple events into complex events. However our usage of the framework is hindered by the lack of documentation on the scripts to be written for deploying the EP-SPARQL queries to ETALIS engine. It is also not very clear whether EP-SPARQL queries can utilise SPARQL 1.1 features, if it can be used to specify queries over event streams comprising of named graphs or RDF datasets and if ETALIS can execute such queries.

4. MODELLING EPCIS EXCEPTIONS

As highlighted in Section 3.1, an exception in an EPCIS governing supply chain process is regarded as an event. While an informal identification of exceptions is available, a formal representation and classification of various kinds of exception that could be possibly generated is missing. We believe such a classification is needed in order to enable a knowledge based approach, that can exploit automated inferencing, towards the notification and handling of the exceptions. A generalisation hierarchy based on the exceptions identified in Section 3.1 is required. Also missing is a conceptual schema for capturing the attributes and relationships of an EPCIS exception that can be further exploited for informing the handlers about the features of the exception. We extend the EEM ontology with an Exceptions module, that provides both the taxonomy and the attributes for exceptions, which we further utilise in our monitoring framework.

Exception handling involves three major activities: detection/identification, notification and handling. The process of notification and handling are dependent on the type of exception that is detected. The EPCIS Exceptions module defines a hierarchy of EPCIS exceptions as illustrated in Figure 4. An EPCIS exception event can be considered as a complex event, composed of primitive events using event composition operators [13]. Supply chains track and trace operations most commonly involve a sequence of business steps as exemplified by the scenario in Section 2. The events associated with the business steps that precede an exception have to be reported as part of the notification. The sequential order in which the events occurred before the exception is of crucial importance in deciding the handling mechanism. Further, as every detected exception must be notified immediately, an exception event must be followed by a notification

[18] `http://purl.oclc.org/NET/ssnx/ssn`
[19] `http://ccs.mit.edu/ph/`
[20] Now obsolete

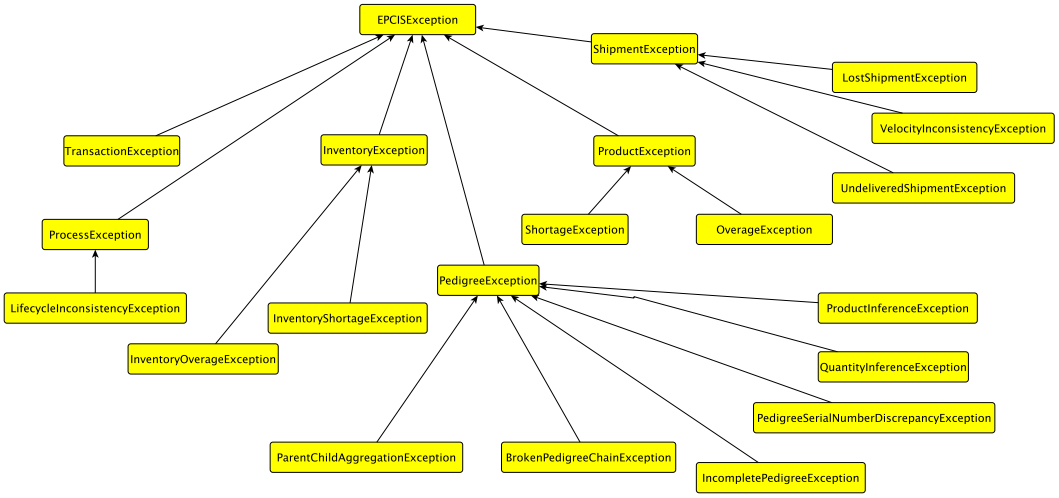

Figure 4: Hierarchy of EPCIS exceptions

event. Exceptions arise as part of a business process executing a specific business step and associated with a specific business process location.

In forward moving supply chains that exchange traceability data via pedigrees, exceptions may arise during the validation of pedigrees received from the upstream partners. For these exceptions, the pedigree that caused the exception needs to be reported. Figure 5 presents a graphical representation of `EPCISExceptionEvent`.

Note that in this paper, we deliberately do not discuss mechanisms for exception handling for two reasons: Firstly, while GS1 has potentially identified the exceptions that may arise during event-based traceability processes, it has not yet defined message exchange choreographies for dealing with these exceptions that could serve as a guide for implementing the exception handling processes. Secondly, even if these choreographies were available, we believe that the handling of exceptions would largely be governed by the contractual obligations defined between the partners.

5. DETECTING EPCIS EXCEPTIONS

As we build on the linked pedigrees based approach for the representation of traceability information, in this paper we are mostly concerned with detecting exceptions that arise when information to be incorporated into the pedigrees is being generated at a stakeholder's end or they are being validated against the received shipment by a downstream partner. As an example, we consider exceptions (e4) identified in Section 3.

Consider the pharmaceutical supply chain depicted in Section 2. As the serialised items, cases and pallets move through the various phases of the supply chain at a trading partner's premises, EPCIS events are generated and recorded at several RFID reader nodes. The event URIs are named graphs, that need to be queried in order to determine the business step and the action taken for each event. In most scenarios the flow of events to the event processor is linear, i.e., we always consider a sequence of ordered[21] events being sent to the event processor. Given the above scenario and the three kinds of events identified in Section 3, exception (e4) missing or incorrect containment hierarchy between items and their containers - `ParentChildAggregationException` may be detected when the commissioned items are being aggregated into cases. The detection mechanism is outlined below.

Step 1: Select EPCs from every `EPCISEvent` where the business step is "packing" and the action is "ADD". Note that since `AggregationEvent` has been defined as subclass of `EPCISEvent`, subsumption reasoning ensures that aggregation events are returned in response to the query below.

```
# Query 1: selects EPCs from an EPCIS event where
# the business step is packing
 SELECT ?epc1 WHERE {
  ?s1 a eem:EPCISEvent;
      eem:hasBusinessStepType ?x1;
      eem:action eem:ADD;
      eem:associatedWithEPCList ?y1.
    ?y1  <http://purl.org/co#element> ?epc1.
    FILTER(contains(str(?x1), "packing"))
  }
```

Step 2: Check if the items that are part of the event have indeed been commissioned in an earlier `ObjectEvent`.

```
# Query 2: checks if all the EPCs which are part
# of the aggregation event have indeed been commissioned
 ASK WHERE {
  ?s2 a eem:ObjectEvent;
      eem:hasBusinessStepType ?x2;
      eem:action eem:ADD;
      eem:associatedWithEPCList ?y2.
  ?y2 <http://purl.org/co#element> ?epc1.
     FILTER(contains(str(?x2), "commissioning"))
 }
```

Results from the above query are sent to the Runtime Monitor (cf. Section 6). When the ASK query returns a "No", a `ParentChildAggregationException` exception is detected.

Step 3: Construct an exception graph and generate a notification event.

[21]Unordered events introduce a different level of complexity which is orthogonal to the focus of this paper.

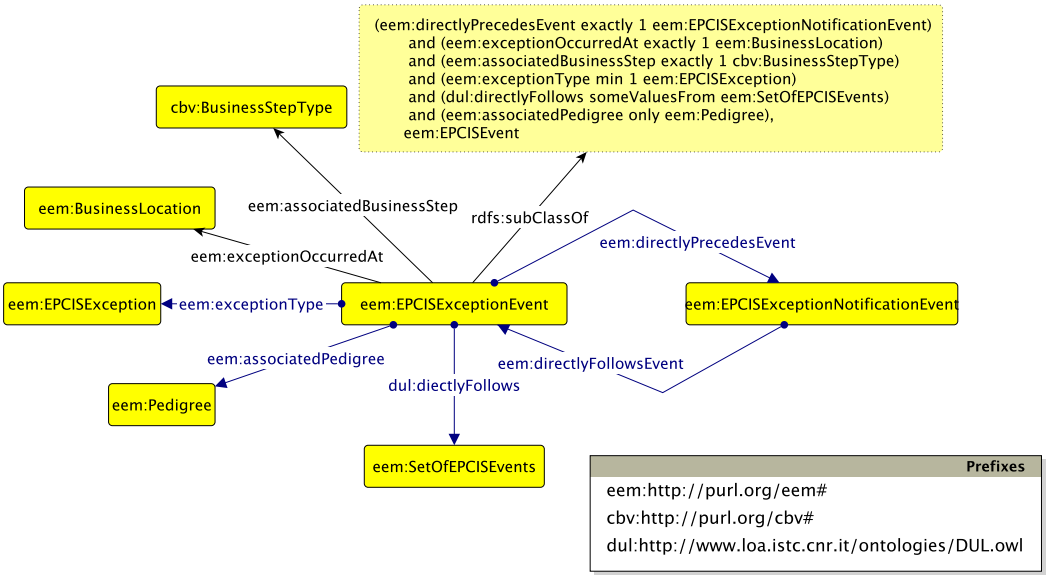

Figure 5: Graphical representation of EPCISExceptionEvent

```
CONSTRUCT
{
 <excepEventURI> a
      eem:ParentChildAggregationException;
      eem:eventOccurredAt "timeLiteral"xsd:datetime;
      eem:associatedBusinessStep cbv:packing;
      ....other triples about the exception
      dul:directlyFollows <SetOfAggregationEvents>.
}
CONSTRUCT
{
 <notificationEventURI> a
      eem:EPCISExceptionNotificationEvent;
      eem:eventOccurredAt "timeLiteral"xsd:datetime;
 eem:directlyFollowsEvent <excepEventURI>.
}
```

Note that if for some reason, the exception is not detected at the manufacturer's end, it can still be detected as exception (e2) - `ProductInferenceException` when the aggregated case reaches the downstream partner who dereferences the pedigrees, extracts the event data and validates the disaggregated items against the received pedigrees.

Note that SPARQL queries for other exceptions can be similarly formulated and executed.

6. REFERENCE ARCHITECTURE

Figure 6 illustrates the workflow and execution environment of our EPCIS exception detection framework. A Java library, LinkedEPCIS[22] for encoding EPCIS events as linked data has been developed. The type hierarchy in LinkedEPCIS is based on the entities defined in the EEM data model. LinkedEPCIS has been built over the Sesame framework[23]. LinkedEPCIS records data about events, which can be persisted as files or dumped in a dedicated EPCIS events triple/quad store.

RFID tag data, read by readers is converted into a stream of linked EPCIS event named graphs using LinkedEPCIS which is deployed on the Edge server or as part of a custom app. The generation of exception and notification events is also an integral part of the library. Every event curated in the event data store, using the library, is systematically assigned an HTTP URI. EPCIS event streams are accessed by the Linked Pedigree server that deploys various components for detecting exceptions and facilitating the generation of linked pedigrees from event IRIs.

The event extractor component extracts the event graphs from the stream and sends them to the monitor component responsible for detecting, notifying and handling exceptions. SPARQL queries as defined in Section 5 are registered within the monitor and executed as the events arrive. For downstream partners the monitor accesses the pedigrees received from the upstream partners to perform the required validations. If exceptions are detected, the monitor constructs the exception graph and sends notifications to the exception handler. If no exceptions are detected, the event graph IRIs, are passed to the Pedigree generator which compiles the pedigrees in accordance to the pedigree definition in [19]. Supply chain models are very commonly analysed based on their simulation results due to track and trace data being commercially sensitive and due to the huge economic investments involved. Companies are wary of introducing new implementation models in their existing supply chain setups. The simulations are however robust, scalable and derived based on a set of real data. The LinkedEPCIS library provides a set of classes that can simulate a pharmaceutical supply chain and generate a large and varied combination of a rich set of EPCIS events including exceptions for each phase and trading partner in the supply chain. Based on the initial results of the simulations, companies can easily incorporate the approach within their commercial IT infrastructure.

[22] `https://github.com/nimonika/LinkedEPCIS`
[23] `http://openrdf.callimachus.net/sesame/2.7/docs/users.docbook?view`

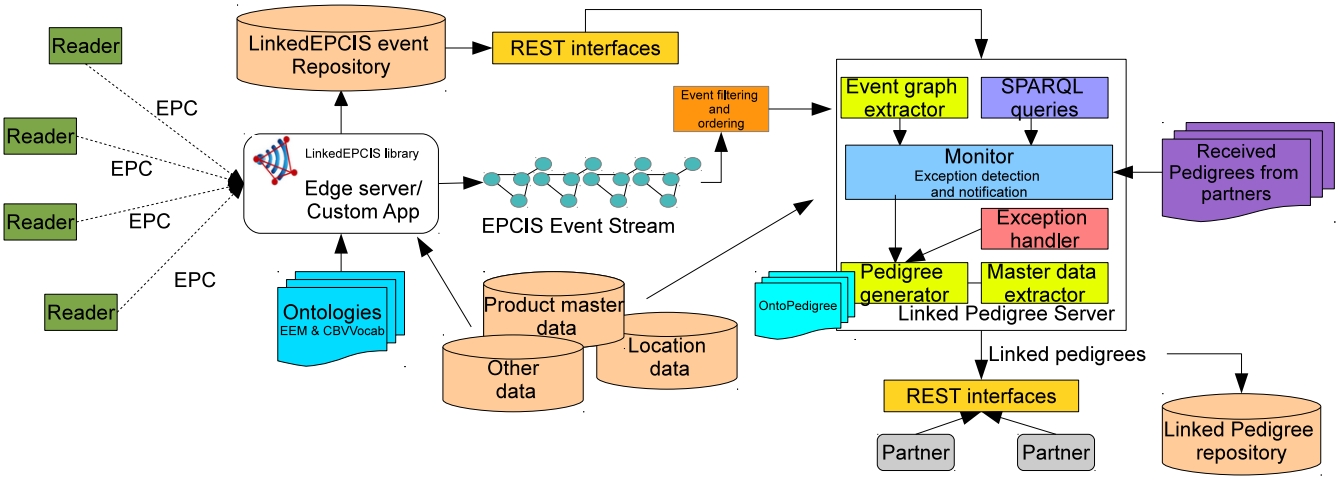

Figure 6: Exception detection and notification Architecture

7. EVALUATION

Our evaluation for the pharmaceutical scenario outlined in Section 2 and the queries outlined in Section 5, focuses on the time taken for the detection of the exception against the number of incorrect or missing EPCs in each `AggregationEvent`.

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, referred to grey literature and interviewed people closely involved in the pharmaceutical sector and EPCIS experts. We referred to a survey [18] 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.

Based on this data we ran a simulation that replicated the volume and velocity of events and generated commissioning and aggregation events, with increasing number of EPCs in the aggregation events that were incorrect or missing from the commissioning events. The incorrect or missing EPCs where randomly introduced in the aggregation events to simulate real world scenarios. The results from the evaluation are illustrated in Figure 7. The evaluations were made on Mac OSX 10.7.5, 2.7GHz Intel core i7, 8GB 1333 MHz DDR3. As observed, the time taken to detect the exception increases sharply with an increase in the number of incorrect or missing EPCs in the aggregation event. On the other hand, varying the number of items in an aggregation from 100 to 500 did not result in any substantial increase in the detection time. Based on our observation we can conclude that for an optimal performance in the detection of exceptions, the number of commissioned EPCs against which the checks are performed need to be suitably optimised by partitioning large volumes of event data, running the checks over each partition and aggregating the results. This can potentially enable quicker detection of exceptions and consequently make a significant impact on the overall efficiency of tracking and tracing within the supply chain. As noted above, changing EPC volumes in the aggregation event itself does not make a significant impact.

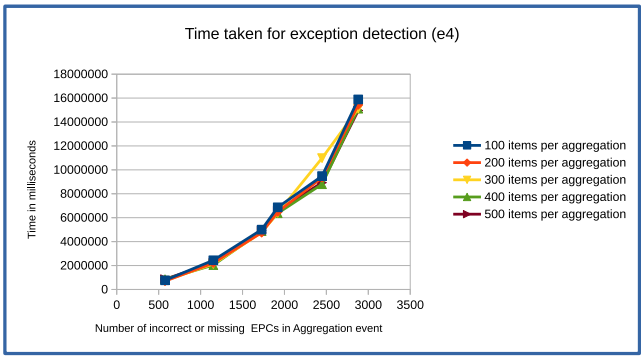

Figure 7: Time taken for exception detection against increasing number of incorrect or missing of EPCs in AggregationEvent

8. CONCLUSIONS

Frequent occurrences of exceptions in supply chain is inevitable due to the complexity of the underlying operations. Their timely detection offers great practical advantages, reduces delays in meeting order fulfillment deadlines and provides huge economical benefits across the supply chain.

We believe that the conceptualisation of supply chain exceptions as events represented as linked data, provides a natural extension to linked pedigrees as the representation of track and trace knowledge. In this paper we have proposed an approach for the modelling and representation of exceptions occurring in supply chains governed by GS1's event-based EPCIS specification and linked pedigrees. We propose a taxonomy of EPCIS exceptions based on typical exceptions identified by the GS1 Healthcare US Secure Supply Chain Task Force. We have extended the EEM ontology with an exception module that provides the minimalistic abstraction needed to construct an exception graph at runtime and notify the business activity responsible for handling the exception. The LinkedEPCIS library provides the API re-

quired for seamlessly incorporating the generation of EPCIS exceptions as linked data in EPCIS track and trace implementation. We have proposed an architecture that incorporates a monitor for detecting the exceptions by exploiting SPARQL queries, constructing the exception graph and generating the appropriate notification event. Finally, we have evaluated and exemplified our approach on stakeholders in the pharmaceutical supply chain.

While we believe that we have the fundamental framework for linked pedigree based exception detection and handling in supply chains, we would like to extend it with a visualisation feature that provides useful supply chain performance metrics based on the business process raising the exception, the frequency and type of exceptions that are observed. This implementation is currently in progress. We are also investigating the utility of other event composition operators in the detection of exceptions, so we can exploit other event processing frameworks for some of the exception detection tasks.

9. ACKNOWLEDGEMENTS

The research described in this paper has been partially supported by the EU FP7 FI PPP projects, SmartAgriFood (`http://smartagrifood.eu`) and FISpace [24].

10. REFERENCES

[1] F. M. Alexander Ilic, Thomas Andersen. EPCIS-based Supply Chain Visualization Tool. Auto-ID Labs White Paper WP-BIZAPP-045, 2009.

[2] J. F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 1984.

[3] 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.

[4] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. Rfid data processing with a data stream query language. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, April 2007.

[5] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05. ACM, 2005.

[6] Davide Francesco et al. C-SPARQL: a Continuous Query Language for RDF Data Streams. *Int. J. Semantic Computing*, 2010.

[7] Della Valle, E. et al. It's a Streaming World! Reasoning upon Rapidly Changing Information. *Intelligent Systems, IEEE*, 2009.

[8] C. Dellarocas and M. Klein. A knowledge-based approach for designing robust business processes. In *Business Process Management, Models, Techniques, and Empirical Studies*, London, UK, UK, 2000. Springer-Verlag.

[9] M. Garofalakis, J. Gehrke, and R. Rastogi. *Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[24] `http://www.fispace.eu/`

[10] B. N. Grosof and T. C. Poon. Sweetdeal: Representing agent contracts with exceptions using xml rules, ontologies, and process descriptions. pages 340–349. ACM Press, 2003.

[11] M. N. Huhns, L. M. Stephens, and N. Ivezic. Automating supply-chain management. In *AAMAS*. ACM, 2002.

[12] Le-Phuoc, Danh et al. A native and adaptive approach for unified processing of linked streams and linked data. In *Proceedings of the 10th International Conference on The Semantic Web*, ISWC'11. Springer-Verlag, 2011.

[13] Y. Liu and D. Wang. Complex Event Processing Engine for Large Volumes of RFID Data. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, 2010.

[14] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[15] A. Özkohen and P. Yolum. Predicting exceptions in agent-based supply-chains. In *Proceedings of the 6th International Conference on Engineering Societies in the Agents World*, ESAW'05, pages 168–183, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] Rinne, Mikko et al. Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In *OTM Conferences (2)*, Lecture Notes in Computer Science. Springer, 2012.

[17] Schapranow, Matthieu-P. et al. A Formal Model for Enabling RFID in Pharmaceutical Supply Chains. In *Hawaii International Conference on System Sciences*. IEEE Computer Society, 2011.

[18] A. Sinha. Systems engineering perspective of e-pedigree system. Master's thesis, Systems Design and Management (SDM), MIT, 2009.

[19] 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.

[20] 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.

[21] M. Solanki and C. Brewster. Modelling and Linking transformations in EPCIS governing supply chain business processes. In *Proceedings of the 15th International Conference on Electronic Commerce and Web Technologies (EC-Web 2014)*, 2014.

[22] F. Wang, S. Liu, and P. Liu. Complex RFID event processing. *The VLDB Journal*, 2009.

[23] Weidlich, Matthias and Ziekow, Holger and Mendling, Jan. Optimising Complex Event Queries over Business Processes Using Behavioural Profiles. In *Business Process Management Workshops*, Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2011.