Bringing Agility into Linked Data Development: *

An Industrial Use Case in Logistics Domain

Pinar Gocebe Oguz Dikenelli
Ege University
gocebepinar@gmail.com
odikenelli@gmail.com

Nuri Umut Kose
BIMAR Information Technologies
umut.kose@bimar.com.tr

Abstract

Logistics is a complex industry where many different types of companies collaborate in order to transport containers to the last point. One of the most important problem in logistics domain is observation and monitoring of container life cycle where each step of the container transportation may be performed by different company. Thus, observing and monitoring of the container's life cycle in real time become a challenging engineering task. In this research, Linked Data development infrastructure has been used to implement dynamic container observation and monitoring system for ARKAS company which is the leading logistics company in Turkey. During the development of the system, it has been observed that agile practices like feature/story oriented development, test first development and usage of Agile Architecture approach improves the product and project management quality. So, a new methodology has been proposed based on these practices for Linked Data development.

Keywords Linked Data Development Methodology, Agile Analytics, Agile Architecture

1. Introduction

Logistics is a complex industry where many different types of roles such as shipping agency company(ies), port/ship operator(s), land, rail and air transportation companies collaborate in order to transport a container to a destination. These companies require a collaboration infrastructure in order to resume the whole transportation process seamlessly. This collaboration infrastructure should provide an integration environment for the information systems of the companies in order to manage all sub-transportations and needs to be open in a sense that it should to be easy to be able to add/remove company(ies) into the process. In addition to the integration of the information systems, monitoring of the process is also critical for the effective management of the process. For instance, important events like delay in the completion of the land transport, starting of the discharging operation at the port can be very critical for the roles that are participated in the process.

This paper introduces an implemented architecture based on Linked Data infrastructure for the well known logistics problem "Observation and Monitoring of Container Life Cycle". Application is developed within the ARKAS Holding which is one of Turkey's leading logistics and transportations company. It operates in different fields such as sea, land, rail, air transportation, ship operations and port operations. Therefore, executing "Observation and Monitoring of Container Life Cycle" problem in real time is a challenging engineering task since all sub-transportations may run in parallel on different software systems of different companies. The end goal is to have managers and customers be able to track the container transportation life cycle.

In the last decade, EDI-based standards (EDIFACT, RosettaNet, STEP, AnsiX12), XML standard and Service Oriented Architecture (SOA) approaches are used for solving the integration problems of logistics industry[1, 2]. These standards provide common syntax for data representation. SOA provides an application integration infrastructure between different companies via web services. In the EDI-based standards, messages are pushed among the organizations on a predefined time and these messages are translated into suitable format for receiver or sender organization in order to provide communication. However, these technologies are not sufficient to ultimately solve the integration challenges in large enterprises. In the SOA approaches, the most important problem is connectivity [3]. Identifiers (ID) of the data which are stored in the database are acknowledge inside of the systems and they lose their meaning in the other systems. Finding the operation of a web service that will be called by the identifier is constructed into the software application logic. This is elaborated application logic when considering the complexity of the logistics industry. EDI-based standards are not suitable for real-time applications since data may be outdated when information is updated and this is not directly send in an message. Also, too many conversion is needed when organizations are used different types of EDI formats.

Linked Data infrastructure seems like an appropriate technical solution for the requirements of the logistics industry, since it provides an integration environment which is more flexible, extensible and open to the exterior when necessary. Linked Data standards and infrastructure are prevalently used to integrate enterprise information and business processes[4–6]. In Linked Data based integration, identifiers (URI) is not only known in system-wide, but also known in web-wide and the data which is represented with these URIs is reachable from HTTP protocol. Thus, all data sources within the company and/or on the web can be connected with each other creating a huge knowledge base and software systems can use this knowledge base independently from each other. Therefore, Linked Data technologies propose a new solution to dynamic, dis-

* This work is funded by the Republic of Turkey Ministry of Science, Industry and Technology

WWW2015 Workshop: Linked Data on the Web (LDOW2015).

tributed and complex nature of the "Observation and Monitoring of Container Life Cycle" problem specifically and logistics domain in general.

During the development of the aforementioned "Observation and Monitoring of Container Life Cycle" application, the development team defined a development methodology. Since, the development team has a long time experience in Agile Development, the proposed methodology brings agile practices into the Linked Data development. The Methodology called as BLOB(A Methodology to **B**ring Agility into **L**inked **O**pen Data Development for **B**usinesses) has evolved through the iterations of the development. Its final version which is overviewed within the paper has following contributions:

- Feature/story oriented Linked Data development.
- Introducing Agile Architecture[7] approach to Linked Data development.
- Applying test first approach to Linked Data development.

At the moment, "Observation and Monitoring of Container Life Cycle" application is operational and tested by customer operation unit of ARKAS holding. The paper introduces how the application and its architecture evolved through the iterations of the proposed methodology.

2. The Problem : Observation and Monitoring of Container Life Cycle

In the logistics industry, customers' loads are transported to a final destination from a start location within containers. Through this transportation, containers are proccessed in a variety of work areas such as port, warehouse, land, rail and air. For instance, let us consider a company that wants to send its load to customers in Munich/Germany from Manisa/Turkey. This company is aggreed with a forwarder company for the transportation. This transportation is planned as a four-stage; a land transportation from Manisa to Izmir Alsancak port, a maritime transportation from Izmir Alsancak port to Piraeus port of Athens, a maritime transportation from Pireaus to Venice port and a land transportation from Venice port to Munich. Also, it takes approximately 10 days and there is not an interface to get information about the transportation.

Throughout the transportation, customers want to learn exact status and position of the transported loads and this problem is dealed in wireless network and RFID studies[36, 38, 39] from the viewpoint of hardware. But, these studies is related with only position and status of the cotainers. They are not interested with links of container with other concepts in the domain. In our case, customers only takes information by calling customer operation unit of ARKAS. All transportation companies use their special information technologies and infrastructures. Furthermore, there is not an integration environment between the systems of these companies. Any latency in the transportation process is affected all other related transportations. Therefore, transportation must be constantly monitored by directly calling companies in charge and this brings too much operational load to customer operation unit employees. In this research, we aim to solve the "Observation and Monitoring of Container Life Cycle" problem of the logistics industry by using Linked Data infrastructre. For this purpose, following two main requirements of the industry are performed;

1. Providing an Integration Environment

   Transportation history of the container which is distributed into the different software systems should be integrated.

2. Monitoring Containers

   Events of the container transportation in different work areas should be monitoring and customers should be informed about transportation status.

3. Overview of the Methodology

It is well understood that agility is the good approach to cope with changing business and architectural requirements in software development. Not only software development, but also business intelligence and analytics implementations benefit the agile style of development as proposed in [8]. The proposed methodology takes some practices from agile software development, agile analytics like Scrum[9] and XP[10] such as feature/story oriented development, test first approach[11], customer involvement and continuous integration[12]. Scrum infrastructure is used to managed project with self-organized team dynamics and iterative life cycle. Linked Data environment changes constantly by occurenses of new data sources, new links and changes in ontologies. This highly dynamic environment may cause changes in business and architectural requirements. Thus, agile practices used within the linked data methodoloy makes the methodology suitable for highly dynamic environment of Linked Data application development.

Linked Data development is very young domain where critical tools and architectural patterns are constantly evolving. Also, changes in business requirements and/or Linked Data environment may affect the initial architectural assumptions. Thus, development team should observe the evolution and the performance of the architecture throughout the development. Observation of the arhitecture evaluation throughout the methodology is clearly defined in Agile Architecture approach [7]. As defined in Agile Architecture approach, the proposed methodology sets the architectural style and evaluation criterias at the beginning of the each iteration and validates the architectural assumptions at the end of the iteration.

Linked Data development requires some specific tasks as defined in various Linked Data development methodologies[13–16]. Also, Ontology Modelling literature has a long history with many proposed and used methodologies[17–21]. The proposed methodology takes the required tasks from Linked Data development and Ontology Modelling approaches and combines them with agile practices within a iterative life cycle. The methodology is evolved through the four iterations of the application development which took more than a year and each iteration contains small sprints which took between 2 and 4 weeks. During the iterations, it is clearly understood that we need to define test first approach by identifying test case modelling and testing points in the life cycle. Another critical observation is the neccessity of parallel execution of *Linked Data Environment Implementation* and *Application Development* cycles. Linked Data Environment Implementation cycle includes all the activities related with data perspective. On the other hand, Application Development cycle includes software development activities on the top of the generated linked data. In Figure1, inner cycle of the methodology represents the Application Development cycle and the outher one represents Linked Data Environment Implementation cycle.

3.1 Analysis

In the analysis activity, application requirements are identified and managed by product owner by incorporating with necessary project stakeholders like customer role(s) and development team member(s). As a first step of the analysis, main goals of the applications are identified and for each main goal new user stories are defined to satisfy the goal. The critical aspect of the analysis phase from the linked data perspective is the identification of the data sources that require for the user story. The data sources are identified by development team and attached to the "*Application Requirement Card*" (ARC). The second critical differences from the

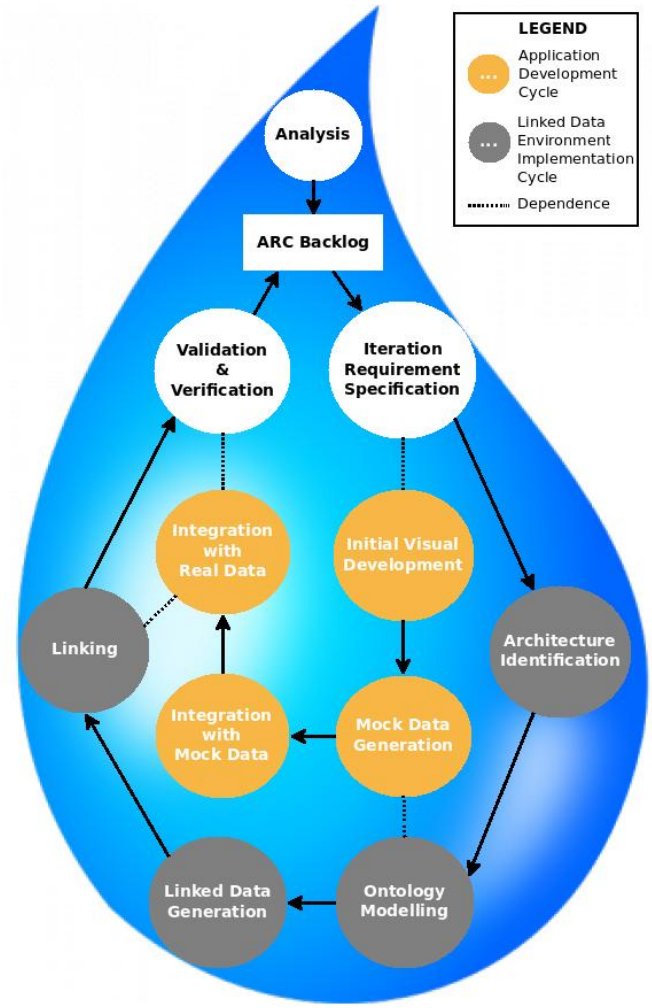

Figure 1. BLOB: A Methodology to **B**ring Agility into **L**inked **O**pen Data Development for **B**usinesses

classical user story definition is the competency question section of the card. Competency questions is the well known approach in the ontology development literature [17, 19–21] in order to limit the scope of the ontology and validate the developed ontology [37]. In our case, competency questions are drived by considering the linked view of the data sources and validation of the user stories through the this linked view. These competency questions are the main sources of the user story validation test cases. Analysis activity is not part of the iterations and can be executed when neccessary. Product owner observes the evolution of the implementation, collaborates with customer constantly and may define new goals, refined existing goals and/or define new ARC(s) for new or existing goals according to situation. These ARC(s) are maintained in the ARC backlog.

The ARCs are defined for each story including the following parts;

- *ID:* Identifier of the ARC
- *Application Goal:* Includes intended use of the application. It lends assistance to draw the boundaries of the application.
- *Selected Data sources the story:* Data sources that will be converted to Linked Data and be consumed by the application. They can be relational databases, documents, web pages etc.
- *User Stories:* Senarios of use from the viewpoint of end-users in order to implement the defined feature of the goal.
- *Competency Questions:* Questions to validate the story.

3.2 Iteration Requirement Specification

Iteration Requirement Specification (IRS) is a planning activity. ARCs that are maintained in the ARC backlog are prioritized according to their business value and included data sources. Considering the data sources in story prioritization is critical in terms of Linked Data development. If more than one sources are included in the story, this situation affects the Linked Data Generation, Linking and also Architecture Identification activities. Properties of core Linked Data architecture depends on publishing and integration decisions on different data sources. Thus, inclusion of more than one data source is critical in terms of establishing and evaluating core architecture in early iterations. Thus, at the end of the IRS activity development team decides the stories for the iteration depending on the business and architectural perspective.

3.3 Linked Data Environment Implementation Cycle

3.3.1 Architecture Identification

Architecture Identification activity is first step of the architectural agility. Firstly, user stories of the ARC(s) that are selected in the previous activity are analyzed to identify the architectural pattern(s) depending on the architectural requirement of the application. From test first perspective, test planning for architecture evaluation are defined in this activity. Architecture evaluation is conducted by two levels of testing. The first level focuses on the retrieving performance of generated Linked Data and the other level focuses on the evaluation of selected quality attibutes[22] like performance, scalability, availability etc for the final application. The first level is applied in the Linked Data Generation and/or Linking activity and second level is applied in Validation&Verification activity. At this point, data retrieving criterias, quality attributes and their expected boundaries are identified and documented as initial architecture evaluation test plan.

There are three well known architectural patterns for consuming Linked Data according to application requirements: the On-The-Fly Dereferencing pattern, the Crawling Pattern and the Query Federation Pattern [23]. On-The-Fly Dereferencing pattern conceptualizes the web as graph of documents which contains dereferanceble URIs. Thus, an application executes a query by accessing a RDF file by dereferencing the URL address then follows the URI links by parsing the received file on-the-fly[24]. In the Crawling Pattern, web of data is constantly crawled by dereferencing URLs, following links and integrating the discovered data on the local site. Query Federation Pattern is based on dividing a complex query into subqueries and distributing sub-queries to relevant datasets. Query federation requires accessing datasets via SPARQL endpoints in order to execute sub-queries on distributed data sources.

All of them have disadvantages and advantages while making architectural decision should take into account. In On-The-Fly Dereferencing pattern, complex operations are very slow because of dereferencing thousands of URIs in the background, but stale data is never processed. The main advantage of the crawling pattern is performance. Applications can use high volume of integrated data in much higher performance than other patterns. On the other hand, the main disadvantage of this pattern is data staling and complexity of automatic linking of data on the fly. Query Federation pattern enables applications to work with current data without needing to replicate complete data sources locally. On the other hand, the main problem of this pattern is performance of the complex queries especially when query needs to join data from large number of data sources.

Also, there are a wide range of Linked Data Design Patterns for modelling, publishing and consuming in the literature [25]. Development team or data publishers can be use appropriate pattern or mixture of these patterns. However selection of these patterns is related with different factors that affect architectural decisions such as number of data sources, data freshness level, application response time and ability to discover new sources at runtime. Development team makes architectural identification decision(s) based on the selected quality attributes and all of these factors.

3.3.2 Ontology Modelling

In the Ontology Modelling activity, concepts of domain and relationships between these concepts are modelled and implemented in a formal language. Ontology Modelling activity is inspired from the following literature [17–21, 26–28]. This activity is composed of *Conceptualizing, Ontology Implementation, Integration/ Modularization* and *Test Case Generation sub-activities*. Figure 2 shows that ontology modelling lifecycle.

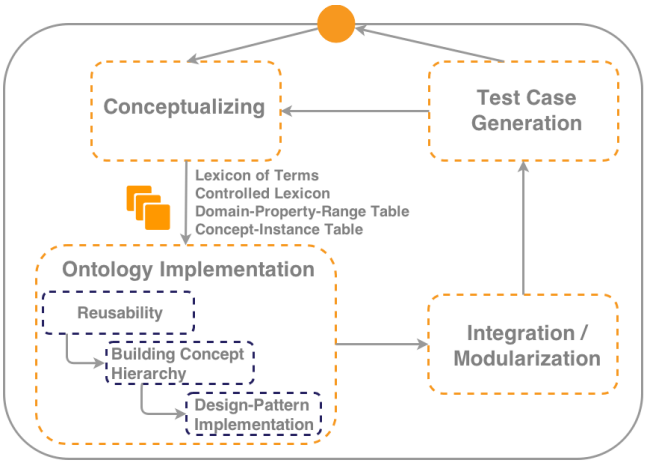

Figure 2. Ontology Modelling Life Cycle.

Conceptualizing

Conceptualizing is a common activity in any ontology modelling methodology[17–19]. Main goal of the *Conceptualizing* activity is construction a conceptual model of domain knowledge. Domain experts and ontology engineers analyze data sources of ARC(s) which are selected in the IRS phase and prepare a rough list of terms, then the terms which are out of the goal scope that belongs to the selected ARC(s) are eliminated. Ontology engineers and domain experts prepare a *Lexicon of Terms* document that contains a list of domain concepts and then create a *Controlled Lexicon* with explanations of the concepts. These explanations lend to find additional concepts. Also, *Domain-Property-Range* and *Concept-Instance* tables are prepared in order to simplify following ontology modelling sub-activities. The former table represents relationships between source and target terms and the latter represents instances of concepts.

Ontology Implementation

Ontology Implementation activity defines a formal ontology model from the defined conceptual model and implement it with a ontology modelling language. First sub-activity of the Ontology Implementation step is *Reusability*. Reusability aims to reuse known and accepted ontologies on the web. Ontology engineers try to find an ontology in semantic web search engines such as Swoogle[1], Sindice[2], Watson[3] and so on. If there is a ontology that overlaps outputs of conceptualizing activity, this ontology is taken as input to the following activities.

If there is not a suitable ontology, *Building Concept Hierarchy* sub-activity is taken place. In this activity, ontology engineers validate taxonomies of terms in the Domain-Property-Range Table according to OntoClean Methodology [26]. After the hierarchy validation, *Design-Pattern Implementatio*n sub-activity uses the ontology design patterns in the literature[27] and structure of the ontology is improved based on the selected pattern(s). Finally, the ontology is implemented with a formal language such as RDFs, OWL by using capabilities of an ontology development environment. For example, TopBraid Composer[4], Protege[5], WebProtege[6].

Integration/ Modularization

Main goal of the Integration/ Modularization activity is achieving reuse, maintainability, and evolution for large ontologies. Inspiring from the ANEMONE[28], we examined the conceptual links between concepts which are generated in the previous iterations and the concepts of this iteration. After that, ontology is divided into modules or common concepts of modules are integrated into a new ontology module.

Test Case Generation

Ontology Modelling activity focuses on the data sources that defined in the ARC(s) and generates the metadata part of the ontology for the focused data source. This activity also identifies linking requirement(s) between the ontologies in metadata level. At this point, it is possible to define test cases in the ontological level to validate consistency and competency of the developed ontologies and the linking requirement(s). The main source to define test cases is the competency questions that defined in *Analysis* activity. These questions are refined based on the knowledge of the developed ontologies and linking requirement(s). Also, new competency questions may be added , if they are needed. These competency questions are transfered to the real SPARQL queries to validate that developed ontology satisfies the execution of the competency questions. These queries are saved as an ontological test cases for the ARC(s) at hand.

3.3.3 Linked Data Generation

Linked Data generation activity is related to generate linked data from selected data sources according to the ontology model(s). The data generation process differs according to being whether data sources structured (e.g. databases), semi-structured (e.g. XML, XLS, CVS, etc.), or un-structured (e.g. HTML, XHTML, etc.).

Structured data is mapped directly to the ontologies via RDB2RDF convertors as D2RQ[7] or Ultrawrap[8]. Also, "*RDB2RDF Mapping Patterns*"[29]can be used in creation of R2RML[9] mappings that are used by RDB2RDF converters. Semi-structured data sources are proccesssed by using toolsets such as tripliser[10] or Google Refine RDF Extension[11] that allow to convertion according to par-

[1] http://swoogle.umbc.edu/

[2] http://sindice.com/search

[3] http://watson.kmi.open.ac.uk/WatsonWUI/

[4] http://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/

[5] http://protege.stanford.edu/

[6] http://protegewiki.stanford.edu/wiki/WebProtege

[7] http://sw.cs.technion.ac.il/d2rq/tutorial

[8] http://capsenta.com/#section-ultrawrap

[9] http://www.w3.org/TR/r2rml/

[10] http://daverog.github.io/tripliser/

[11] http://refine.deri.ie/

ticular instructions. Except for RDF converters, Natural Language Processing (NLP) methods such as tagging can be used to acquire data from un-structured data sources.

Test case(s) to validate the ontologies are generated in the previous activity (Test Case Generation sub-activity of the Ontology Modelling). At this point, instance data is generated for the developed ontologies. Thus, it is possible to verify that real data sources are correctly transformed to the ontological instances. For this purpose, test automation script(s) that uses generated test case(s) (SPARQL queries) are written to verify that expected results is equals to the results of the SPARQL queries and these script(s) are included into Continuous Integration (CI) infrastructure.

3.3.4 Linking

One of the most important principles of Linked Data is Linking. In this activity, connections are established between data sources in manually or automatically. For this purpose, SILK[12] and LIMES[13] link discovery frameworks can be used in order to automatise this process. If unstructured sources like text will be linked, tools such as Dbpedia Spotlight[14] can be used. Also, another method is using same URIs at ontological level to establish links manually between sources.

Similar to Linked Data Generation activity, in this activity test automation script(s) of the SPARQL queries that contains linking requirements are written and integrated into CI infrastructure.

3.4 Application Development Cycle

3.4.1 Initial Visual Development

Linked Data visualization can be a complex and time consuming task depending on the size of the visualized content. So, it is critical to start the application development cycle at the beginning of the iteration. Firstly, a visual interface draft is identified considering the ARC(s) requirements. Then, team evaluates the different layouts of the selected visualization infrastructure(s) and produces new interface design examples for the working ARC(s). These examples are discussed with the customer and then initial visual design is decided for the iteration.

3.4.2 Mock Data Generation

Application does not only include user interface design components, it also includes data integration layer to handle the data coming from the Linked Data side. Real Linked Data is generated at the end of the *Linking* sub-activity of the *Linked Data Environment Implementation* cycle. But, application development should not wait till linked data generation and should continue seamlessly. Thus, development team participate into *Ontology Modelling* activity and/or cooperate with the ontology modelling team to develop mock linked data repository as the ontologies occures. These mock repositories give chance to work on the data integration layer of the visual design and also improve the visual components further.

3.4.3 Integration with Mock Data

In this activity, visual design and data integration layer of the design is completed by using generated mock repositories. Since, all visualization is fully functional development team can start to define accaptence test senarios. Competency questions defined in *Analysis* activity and refined in the *Ontology Modelling* activity are used to shape acceptance test senarios. Since, these senarios are defined considering the implemented visual design, developer team implement visual test automation scripts for each senario and integrate these senarios into CI infrastructure.

3.4.4 Integration with Real Data

In the final sub-activity of application development cycle, visual design and data integration layer of the visual design are connected with real generated Linked Data sources. Final implementation is tested with visual test scripts and whole application becomes ready for Validation&Verification activity.

3.5 Validation & Verification

Visual test automation script(s) and architecture evaluation test plan(s) are inputs of the Validation&Verification activity. All visual application test scripts are evaluated with customer and development team together to validate that these script(s) cover all ARC(s) requirements in the iteration. Then, architecture evaluation test(s) are created according to architecture evaluation test plan(s). The iteration is ended when all the defined tests are passed.

4. Overview of the Methodology Implementation

In the *Analysis* activity, two main goals were identified by stakeholders as *"Monitoring of the Container Transportation"* (monitoring goal) and *"Observation of the Container Transportation History"* (observation goal). During the discussion, it was understood that primary customer for the system is customer operation unit employees. At this point, user stories were generated from the customer perspective. For the observation goal, stories generally focused on knowledge about the sub-transportations in the container transportation life cycle. For instance, a user story was defined as "As a customer, I want to learn history of the specific booking in a known port". When stories were analyzed, it was seen by the development team, many stories required knowledge from different data sources. For example, to drive booking knowledge in a known port, knowledge of the agency and port data sources were needed. For the monitoring goal, stories generally define the monitoring rules like "As a customer, I want to learn specific container is loaded to a specific ship". Thus, main senarios of these goals, their competency questions and data sources were defined as ARC(s) and added to the backlog. Figure 3 is represented an example ARC.

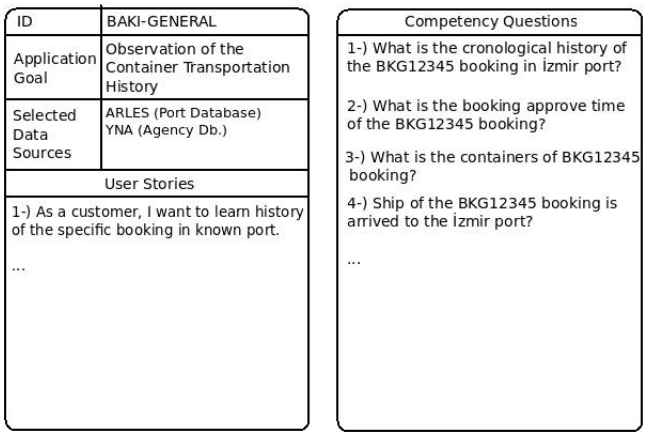

Figure 3. Application Requirement Card for First Iteration.

4.1 Overwiew of Iteration 1

In the *IRS* activity, customer indicated that observation goal is more urgent for their daily operations. Thus, it was understood that ARC(s) of these goal has higher priority. The development

[12] http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/

[13] http://aksw.org/Projects/LIMES.html

[14] https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki

team outlined the criticality of the agency and port data sources, since these sources contains major part of the whole container transportation life cycle. Therefore, ARC(s) included these sources were selected for this iteration.

The development team proceeded to *Architecture Identification* activity and decided to use Query Federation pattern to avoid the stale data problem. Since, agency and port data sources are stored in different relational databases, the development team decided to use RDB2RDF tool for Linked Data generation as seen in Figure 3. After architecture evaluation test plan(s) were defined, federated query response time (max 0.5 second) was taken as a base criteria to evaluate retrieving performance of generated Linked Data. Also, a load test (max 1second response time for each concurrent 20 users) was planned to evaluate performance and scalability of the final application.

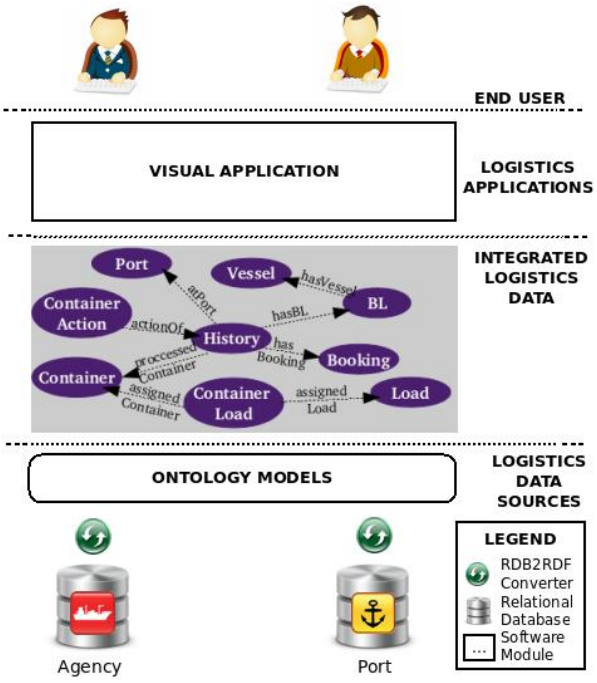

Figure 4. Architecture of First Iteration.

Ontology Modelling activity started after the *Architecture Identification*. After the initial conceptualizing effort, the development team began to search logistics ontologies that proposed in the literature. One of these studies aims to plan military transportation domain with different transportation types[30]. However, this study have not scrutinize intimately the transportation field and focuses on planning and scheduling. In study [31], events and actions that are occured in the logistics processes of enterprises are defined, but transportation process is not detailed. In[32–34], an OWL ontology for the formal representation of logistics services defined and OWL-S usage is examplified for service oriented logistics operations. iCargo[15] and CASSANDRA[16] projects are tried to integrate data which comes from different sources and whereupon requirements such as transportation risk assessment and energy conservation are performed[35]. However, the proposed ontologies is not available for public use. Thus, the development team decided to develop required ontologies from scratch. To this end, in this step the ontologies which represent port and agency data sources were generated by applying the previously defined ontology modelling activities.

In the *Linked Data Generation* activity, data sources were converted into RDF format by using an RDB2RDF tool according to developed ontologies and published in a local server. These convertions were defined in R2RML language by applying mapping patterns in [29]. After the Linked Data generation of port and agency sources, the development team immediately started to *Linking* activity(without any testing of individual sources). Since, there are unique fields in the agency and port systems such as booking, container and bill of lading numbers, we did not use an automatic link discovery tool. Linking of these sources were done by making URIs of them same in ontological level.

Since, reponse time of the federated query execution was selected as Linked Data generation evaluation criteria in *Architecture Identification* activity. The development team started to rework on competency questions of the selected ARC(s), revised existing ones and added new questions. Then, these questions were converted to SPARQL queries (some minor changes required in ontology model) and executed over the generated linked data. Unfortunately, architecture did not satisfy the identified performance limit in the *Architecture Identification* activity. The development team tried to improve query response time by using different toolsets such as Ultrawrap, D2RQ and Oracle Spatial and Graph[17]. Also, ontology model(s) were changed to simplfy the complex mapping(s). But, query performance was still away of expected 0.5 second and ontology structure became unrealistic from domain perspective. Thus, the development team decided to terminate the iteration and began to new one.

4.1.1 Lesson Learned

- Since, learning of Linked Data development knowledge set takes time and prevent the smooth flow of activities within the iteration, plan an education for the development team about the Linked Data technologies including ontology modelling, R2RML mappings, SPARQL (as a minimum set).
- Testing of Linked Data generation is conducted at the end of Linking activity. Some SPARQL queries require changes in ontologies and causes to return Ontology Modelling activity again. So, as a general rule develop test cases and execute tests when neccessary knowledge is ready. In this case, comptency questions can be revised and SPARQL queries can be generated at the end of the Ontology Modelling activity (methodology cycle was revised according to this observation). Also, seperate these test cases for single data source(s) and federation in Ontology Modelling activity and apply them in Linked Data Generation and Linking activities seperately.
- Integrating architecture identification and evaluation of the identified architecture within the development life cycle is a good idea.

4.2 Overview of the Iteration 2

After experiencing the limitations of the Query Federation Pattern, the development team started to this iteration with a clear goal of changing the Linked Data architecture. Also, the development team selected a new ARC for observation goal which includes warehouse data source. This selection aimed to implement whole methodology life cycle and improve experiences and apply the learned lessons. In the *Architecture Identification* activity, the development team decided to use Crawling Pattern which crawls internal data sources of the company to solve the observed performance problems. At this point, solving stale data problem of the Crawling Pattern became

[15] http://i-cargo.eu/

[16] http://www.cassandra-project.eu

[17] http://www.oracle.com/technetwork/database/options/spatialandgraph

a problem. For this purpose, the development team decided to use a Change Data Capture(CDC)[18] tool and transforming each event that comes from CDC tool to the RDF instances of the ontologies. The development team decided to choose high performance commercial CDC tool and it was planned to develop a Integration Module for transforming CDC events. Also, an open source queue implementation was selected to synchronize the CDC tool and Integration Module. It was also decided to store generated RDF in a central RDF store. General architectural view of this iteration is shown in 5. There was no change in the evaluation criteria of the architecture. Thus, architecture evaluation test plan was kept same.

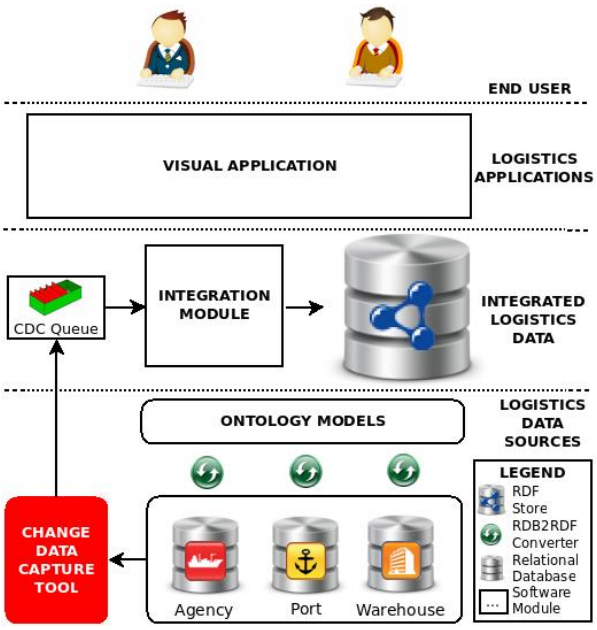

Figure 5. Architecture of Second Iteration.

In the *Ontology Modelling* activity, the development team started to define warehouse ontology model considering previous experiences. Thus, test cases were generated at the end of this activity. The development team revised and extended competency questions of the selected ARC and tried to write SPARQL version of these questions in order to validate adequateness of the generated ontology model. After the successfully defining SPARQL queries of competency questions, the development team is started to *Linked Data Generation* activity. In this activity, R2RML mapping of the warehouse source was created and all converted RDF is stored to central RDF store. Then, generated warehouse data was tested by using generated test cases. Also, agency and port data were converted using previously defined mappings and stored in the central RDF store. In order to solve stale data problem, the development team started to implement an Integration Module. In this module, CDC tool handles changes that occur in the internal relational data sources simultaneous without overload. It sends each change to a java message queue (CDC Queue) using a pre-defined XML format. Integration Module is responsible for consuming XML change messages from CDC Queue. Change messages are converted to RDF triples and RDF store is updated according to found triples. However, it was realized that Integration Module should work concurently in a scalable environment, since these data sources produces approximately three million message per day. In order to handle these message traffic, the development team decided to use AKKA infrastructure[19] whis is highly scalable event driven agent system to manage concurrency, parallelism and fault tolerance. Each update actor in the Integration Module is an AKKA agent which handles XML messages in the queue and converts them to the RDF triple and updates RDF store. AKKA infrastructure of the Integration Module is represented in Figure 6.

In the *Linking* activity, linking was not required because of the establishing links in the ontological level. Thus, only correctness of the links between warehouse RDF data and previously generated RDF data were tested in this activity.

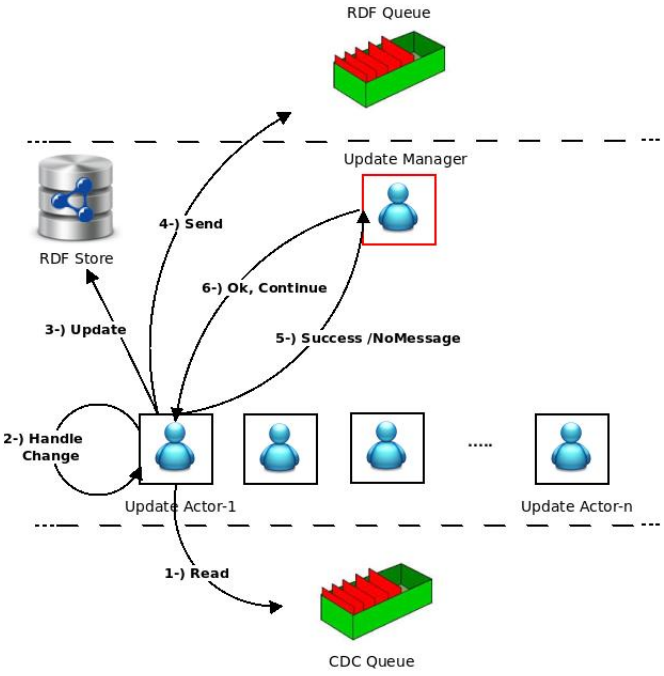

Figure 6. AKKA Infrastructure of the Integration Module

After the successful verification of the generated Linked Data, the development team started to build a visual interface for observation of the container transportation. According to discussions with the customer, the development team identified a visual interface design and select a suitable visualization tecnique and tool. But, implementation of this interface took too long and it was not finished in the planned iteration time limits.

4.2.1 Lesson Learned

- Generation of test cases and making tests in the Ontology Modelling, Linked Data Generation and Linking activities are good idea.
- Implementing a visual interface after the Linked Data generation causes delay in product delivery and decreases motivation of the project stakeholders. Thus, application development should be a parallel process with Linked Data generation (Application Development cycle is added to the methodology).

4.3 Overview of the Iteration 3

In this iteration, the development team selected a new ARC for monitoring goal. Also, an additional ARC was added which includes an external data source (owned by external land transportation company) to the application, since customers wanted to observe history of the transportation life cycle with an external company. In the *Architecture Identification* activity, the development

[18] http://en.wikipedia.org/wiki/Change_data_capture

[19] http://akka.io/

team decided to implement a hybrid architecture which uses Crawling pattern for the internal sources of ARKAS and Query Federation pattern for the external land transportation company, since customer indicated that monitoring events of external company is not a high priority task from business perspective. For handling monitoring in architectural level, team thought to use another AKKA agent organization with separate queue mechanism. In this architecture, change events are transferred to new AKKA organization by the first organization and new organization agent's creates events by applying monitoring rules. Team planned a new load test to verify monitoring with maximum daily messages and decided to observe the real systems' CDC events to formulate the size of the load test.

After the *IRS* activity, the application development team started to design a visual interface for the monitoring goal in the design a visual interface for the monitoring goal in the parallel with *Architecture Identification* activity. The application development team worked on visual designs with the customer. In the *Ontology Modelling* activity, a simple core land transportation ontology was generated which is similar to the relational database schema of external company and test case(s) were generated. Also, the ontology modelling team worked with the application developers for generation of mock data in parallel. The application development team integrated this mock data with visual interface while *Linked Data Generation* activity was implementing. Also, visual acceptance test(s) were generated and added to the CI infrastructure. In the Linked Data Generation activity R2RML mapping of the land transportation source was defined and RDF data of the company published using an endpoint. Also, generated Linked Data was tested by applying previously generated test cases(s) and found errors were fixed. After the Linked Data generation, the application development team integrated visual application with real generated Linked Data and updated visual test(s).

In this iteration, the application was shaped around to notify user when related transportation events are occurred and represent external land transportation history in addition to previously implemented user interface. In order to catch transportation events the development team began to implement the Monitoring Module which includes the new AKKA organization. Monitoring Module responsible for catching all events in the transportation life cycle according to changed RDF triples that are explored by Integration Module. In this module, monitoring AKKA actors try to find affected transportation rules according to changes and serve them to user interface. User interface notifies users about transportation status in real time. Each change may be matched with multiple rules. Final architecture of the application is represented in Figure 7. At the end of the iteration developed application verified in *Validation&Verification* activity. Acceptance test(s) and load test(s) were evaluated with customer and product owner and iteration was finished successfully.

4.3.1 Lesson Learned

- Integration of an external source(s) should be implemented in the early iterations of the application, since it affects architectural decisions and requires lots of effort to introduce linked data infrastructure in a new company.

5. Conclusion and Future Works

Currently "Observation and Monitoring of Container Life Cycle" application is used by customer operation unit of ARKAS holding. At the end of three iterations, the developed application has generated ~300 millions RDF triples by integration agency, port and warehouse data sources of ARKAS. Also, the application handles ~3 millions change messages per day. Customer operation unit test observation and monitoring features of the application with 20 concurrent users (number of the customer operation unit employees). Observation of container life cycle features can be used by container, booking and bill of lading unique numbers. For instance, Figure 8 shows history of a booking whose number is "B14170741" in tree format (This number is correspondence of the resource whose URI is "http://data.arkas.com/general/booking/B14170741" and it is same in the RDF data of each participant company). Moreover, the development team is comfortable to work with BLOB methodology and ready to use it for other linked data project(s).

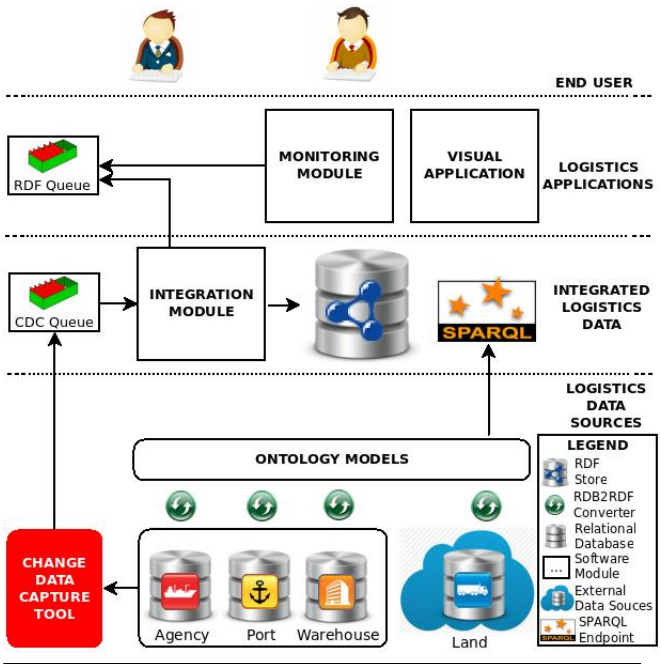

Figure 7. Architecture of Third Iteration.

According to the feedbacks of the employees, It is realized that new ARCs are necessary to visually define new transportation rules for monitoring of the transportation life cycle. In this interface, customers can add/remove/update transportation rules that they want to follow. Also, role based access management is needed, since each customer is interested in different information about the transportation. In the future of the project, these requirements will be implemented.

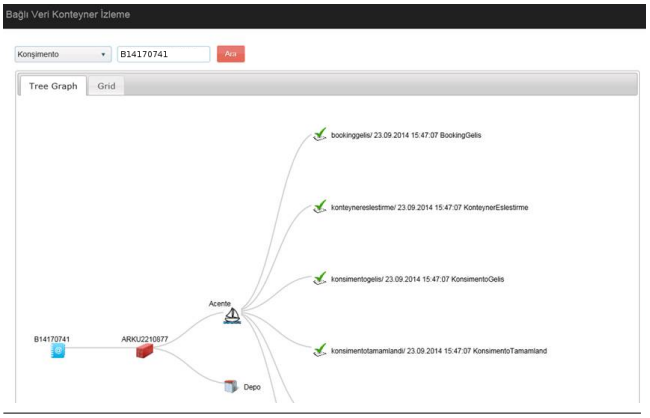

Figure 8. User Interface.

6. Acknowledgement

The authors wish to thanks Bimar Information Technologies Services, ARKAS holding and their employees Guner Mutlu, Necmi Sentuna, Gokhan Daghan, Tugkan Tuglular, Burak Bilyay and Kubra Parkın for their helps. Also, we wish to acknowledge Galaksiya Information Technologies and Consultancy and its manager Erdem Eser Ekinci for their contributions to the developed methodology.

References

[1] Harleman, R. Improving the Logistic Sectors Efficiency using Service Oriented Architectures (SOA). In 17th Twente Student Conference on IT, 2012.

[2] Nurmilaakso, J.M. Adoption of e-business functions and migration from EDI-based to XML- based e-business frameworks in supply chain integration. International Journal of Production Economics 113(2), 721-733, 2008.

[3] Loutas, N. Case Study: How Linked Data is transforming eGovernment. European Commission ISA Programme, 2013, Available at: http://joinup.ec.europa.eu/community/semic/document/case-study-how-linked-data-transforming-egovernment.

[4] Frischmuth, P., Klímek, J., Auer, S., Tramp, S., Unbehauen, J., HolzweiSSig, K. and Marquardt, C.M. Linked Data in Enterprise Information Integration. Semantic Web – Interoperability, Usability, Applicability an IOS Press Journal, 2012.

[5] Mihindukulasooriya, N., Garcia-Castro, R. and Gutiérrez, M.E. Linked Data Platform as a novel approach for Enterprise Application Integration. In the Prooceedings of the 4th COLD Workshop, 2013.

[6] Hu, B. and Svensson, G. A Case Study of Linked Enterprise Data. In Proceedings of the 9th International Conference on The Semantic Web, 2010.

[7] Brown, N., Nord, R., Ozkaya, I. Enabling Agility Through Architecture. Software Engineering Institute, 2010.

[8] Collier, K.W. Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing (1st ed.). Addison-Wesley Professional, 2011.

[9] Schwaber, K. Agile Project Management with Scrum. Microsoft Press, Redmond, WA, USA, 2004.

[10] Beck, K. and Andres, C. Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley Professional, 2004.

[11] Fraser, S., Beck, K., Caputo, B., Mackinnon, T., Newkirk, J. and Poole, C. Test driven development (TDD). In Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering (XP'03), Michele Marchesi and Giancarlo Succi (Eds.). Springer-Verlag, Berlin, Heidelberg, 459-462, 2003.

[12] Campos, J., Arcuri, A., Fraser, G. and Abreu, R. Continuous test generation: enhancing continuous integration with automated test generation. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE '14). ACM, New York, NY, USA, 55-66, 2014.

[13] Auer, S., Bühmann, L., Dirschl, C., Erling, O., Hausenblas, M., Isele, R., Lehmann, J., Martin, M., Mendes, P. N., Van Nuffelen, B., Stadler, C., Tramp, S. and Williams, H. Managing the life-cycle of linked data with the LOD2 stack. In Proceedings of the 11th international conference on The Semantic Web - Volume Part II (ISWC'12), Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, and Jérôme Euzenat (Eds.), Vol. Part II. Springer-Verlag, Berlin, Heidelberg, 1-16, 2012.

[14] Villazon-Terrazas, B., Vilches-Blazquez, L., Corcho, O. and Gomez-Perez, A. Methodological guidelines for publishing government linked data linking government data. Linking Government Data, 27–49, 2011.

[15] Hyland, B. and Wood, D. The Joy of Data - A Cookbook for Publishing Linked Government Data on the Web. Linking Government Data, 3-26, 2011.

[16] Hausenblas, M. Linked Data Life Cycles. 2011, Available at: http://www.slideshare.net/mediasemanticweb/linked-data-life-cycles.

[17] Fernandez-Lopez, M., Gomez-Perez, A. and Juristo, N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In Proceedings of the AAAI Spring Symposium Series on Ontological Engineering, 1997.

[18] Pinto, H.S., Tempich, C. and Staab, S. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In Proceedings of the 16th European Conference on Artificial Intelligence ECAI, 2004.

[19] De Nicola, A., Missikoff, M. and Navigli, R. A Software Engineering Approach to Ontology Building. Information Systems 34(2), 258-275, 2009.

[20] Noy, N. F. and McGuinness, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. 2001, Available at: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

[21] Suarez-Figueroa, M., Gomez-Perez, A. and Fernandez-Lopez, M. The NeOn Methodology for Ontology Engineering. Ontology Engineering in a Networked World, 9-34, 2012.

[22] Meier, J.D., Homer, A., Taylor, J., Bansode, P., Wall, L., Boucher, R. and Bogawat., A. How To - Design Using Agile Architecture. 2008, Available at: http://apparch.codeplex.com/wikipage?title=How%20To%20-%20Design%20Using%20Agile%20Architecture&referringTitle=How%20Tos.

[23] Heath, T. and Bizer, C. Linked Data: Evolving the Web into a Global Data Space (1st edition). Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1, 1-136. Morgan & Claypool, 2011.

[24] Hartig, O., Bizer, C. and Freytag, J.C. Executing SPARQL queries over theweb of linked data. In International Semantic Web Conference, pages 293-309, 2009.

[25] Dodds, L. and Davis, I. Linked Data Patterns. 2012, Available at: http://patterns. dataincubator.org/book/index.html.

[26] Guarino, N. and Welty, C.A. An Overview of OntoClean. Handbook on Ontologies International Handbooks on Information Systems, 201-220, 2009.

[27] Presutti, V., Daga, E., Gangemi, A. and Blomqvist, E. eXtreme Design with Content Ontology Design Patterns. In Proceedings of the Workshop on Ontology Patterns WOP, 2009.

[28] Ozacar, T., Ozturk, O. and Unalir, M. O. ANEMONE: An environment for modular ontology development. Data & Knowledge Engineering 70(6), 504–526, 2011.

[29] Sequeda, J., Priyatna, F. and Villazon-Terrazas, B. Relational Database to RDF Mapping Patterns. In Proceedings of the Workshop on Ontology Patterns WOP, 2012.

[30] Becker, M. and Smith, S. F. An ontology for Multi-Modal Transportation Planning and Scheduling. Carnegie Mellon University Technical Report, 1997.

[31] Lian, P., Park, D. and Kwon, H. Design of Logistics Ontology for Semantic Repre- senting of Situation in Logistics. In the Second Workshop on Digital Media and its Application in Museum & Heritages, 432-437, 2007.

[32] Hoxha, J., Scheuermann, A. and Bloehdorn, S. An Approach to Formal and Semantic Representation of Logistics Services. In Proceedings of the Workshop on Artificial Intelligence and Logistics (AILog) at the 19th European Conference on Artificial Intelligence (ECAI), 2010.

[33] Scheuermann, A. and Hoxha, J. Ontologies for Intelligent Provision of Logistics Ser- vices. In Proceedings of the 7th International Conference on Internet and Web Applications and Services, 2012.

[34] Preist, C., Esplugas-Cuadrado, J., Battle, S.A., Grimm, S. and Williams, S.K. Auto- mated Business-to-Business Integration of a Logistics Supply Chain Using Seman- tic Web Services Technology. In Proceedings of the 4th International Conference on the Semantic Web, 2005.

[35] Dalmolen, S., Cornelisse, E., Moonen, H. and Stoter, A. Cargo's Digital Shadow - A Blueprint to Enable a Cargo Centric Information Architecture. In eFreight Con- ference, 2012.

[36] Ahn, S. B. Container tracking and tracing system to enhance global visibility. In Proceedings of the Eastern Asia Society for Transportation Studies, vol. 5, 1719 - 1727 pp, 2005.

[37] Bezerra, C., Freitas, F., Santana, F. Evaluating Ontologies with Competency Questions. Web Intelligence (WI) and Intelligent Agent Technologies (IAT), IEEE/WIC/ACM International Joint Conferences, vol 3, 284-285 pp, 2013.

[38] S.L. Ting, L.X. Wang, W.H. Ip. A study of RFID adoption for vehicle tracking in a container terminal. Journal of Industrial Engineering & Management, Vol. 5 Issue 1, 22 p, 2012.

[39] J. K. Siror, G. Liang, K. Pang, H. Sheng and D. Wang. Impact of RFID Technology on Tracking of Export Goods in Kenya. JCIT 5(9),190-199 pp, 2010.