An Architecture for Situation-aware Smart Logistics

Maria-Eugenia Iacob
Department of Industrial Engineering and Business Information Systems
University of Twente
Enschede, Netherlands
https://orcid.org/0000-0002-4004-0117

Gilang Charismadiptya
CAPE Groep
Enchede, Netherlands
https://orcid.org/0000-0002-7694-4759

Marten van Sinderen
Faculty of Electrical Engineering, Mathematics & Computer Science
University of Twente
Enschede, Netherlands
https://orcid.org/0000-0001-7118-1353

Jean Paul Sebastian Piest
Department of Industrial Engineering and Business Information Systems
University of Twente
Enschede, Netherlands
https://orcid.org/0000-0002-0995-6813

***Abstract*— Disruptions and exceptions are an important source of risks in logistics, as far as the planning of transportation services is concerned. Failing to rapidly react on and handle such events may lead to serious depreciation of the transported cargo and reputation damage. The Internet of Things seems to be the technology capable of providing the tools required to detect exceptions nearly real-time. However, currently, there is little research on how to enhance the detected exceptions with related information from internal or external sources. Furthermore, most exception detection capabilities rely on experience and not much research exist on how to improve the accuracy of using third-party knowledge. In this paper, we propose a reference architecture for situation-aware logistics. The architecture specification follows the key principles derived from an extensive requirements analysis, the state of the art literature, and the ideas promoted by the Industrial Data Space initiative. The proposed architecture has been instantiated and tested by means of a prototype designed for the case of temperature-controlled transportation services.**

Keywords—IoT, Architecture, Logistics, Information System, Information sharing, Exception Detection, Industrial data space.

I. INTRODUCTION

Disruptions and exceptions are an important source of risks in logistics, as far as the planning of transportations services is concerned. Logistics exceptions have been defined as "as anything that changes the planned process" [21]. A logistics exception can happen during goods delivery, warehousing, ordering, transportation planning, transportation execution and data input [21]. Exception detection is critical as it helps mitigating the risks of both the logistics service provider (LSP) and its customers. For example, in the pharma and perishable food industries, the breakdown of refrigeration equipment can cause the transported products to depreciate. Additionally, breakdown of equipment also leads to delay in the delivery of cargo. Delays, in turn, can lead to loss of revenue for the company, and dissatisfied customers. Thus, failing to react and handle such events rapidly may lead to serious depreciation of the cargo and reputation damage.

To address this problem, we need to detect and communicate exceptions as they happen. This would make process resilience possible, through rapid intervention to compensate or mitigate their negative effects. Detection can be based on monitoring and observation of the logistics process in context, i.e., collecting information on process and context variables, and comparing these with expected norm values. Manual processing of observation data would be too time-consuming and error-prone, if we consider that an average size LSP may have a fleet of more than 100 trucks. Therefore, the use of information systems in this case is vital. More specifically, we need situation-aware information systems to automate the collection, and pre-processing of nearly-real-time information, to allow the detection of deviations from expected values, and the reporting, and subsequent decision-making upon such exceptions. The Internet of Things (IoT) seems to be the technology capable of providing the tools required to detect exceptions nearly real-time. Internet of things is defined more precisely by IEEE in 2015. Internet of Things is a network that connects uniquely identifiable "Things" to the Internet. The "Things" have sensing/actuation and potential programmability capabilities. Through the exploitation of unique identification and sensing, information about the "Thing" can be collected, and the state of the "Thing" can be changed from anywhere, anytime, by anything [34]. However, currently, there is little research on how to enhance the detected exceptions with related information from internal or external sources, and how to integrate them into the business process and application landscape of an LSP.

There are several system architectures already described in the literature that focus on collection of data regarding logistics exceptions. For example, [3] proposes a middleware approach to collect real-time information surrounding logistic objects, [4] introduces the Intelligent Cargo Concept that uses IoT and RFID to support detection of logistics exceptions, and [25] specifies the GS1 standard on the EPCIS architecture that enables creation and sharing of event data for the traceability of logistics operations.

However, these low level technical approaches are not sufficient. To be effective, we also need to exploit them through high level integration with enterprise systems and business processes. This is because exceptions change the normal flow of a business process and trigger corresponding special sub-processes whose purpose is to minimize negative contributions to stakeholder goals. This requires a holistic architectural approach that allows, for the domain of logistics, to represent, relate and reason about stakeholders, their goals, goal-supporting processes, process exceptions, and ultimately, information systems supporting exception detection and handling. In this way, we can align a business-driven view (taking internal and external knowledge into account) with a data-driven view (extracting information from collected event data), and design a solution that is both technically possible and useful.

To the best of our knowledge, such a holistic approach does not exist for the logistics domain, and a complete definition of situation-aware information systems together

2325-6605/19/$31.00 ©2019 IEEE
DOI 10.1109/EDOCW.2019.00030

with its business context for logistics is lacking. We argue that an enterprise architecture (EA) approach [31, 32, 33, 34] is suitable to fill this gap, as EA covers all the aspects mentioned earlier, and is also formal enough to support the precise specification of a solution. Thus, the main novel contribution of this paper is the design of a situation-aware smart logistics enterprise architecture (SSLEA) that uses an IoT infrastructure to facilitate the discovery and handling of exceptions during the execution of transportation processes. The goal of the research is to improve logistics services by providing a better and faster response to exceptions, such that risks of company revenue loss and customer dissatisfaction are minimized.

The proposed architecture was developed following the design cycle of the design science research methodology [5] . The remainder of this paper is structured according to the three phases of the design cycle. Section II presents the problem investigation consisting of a stakeholder and requirements analysis. Section III presents a systematic literature survey that has been guided by the elicited requirements. Section IV covers the solution design, resulting in our SSLEA proposal. Section V briefly discusses the development of a SSLEA prototype, while section VI presents the design validation by using the prototype in a case study originating from the fresh and frozen food logistics sector. Finally, in section VII, we conclude the paper with a discussion of the applicability of the research results, and give some directions for further research.

II. Requirements Analysis

All the highly cited literature sources on design science indicate that the first step in any design research is the execution of a thorough analysis of the goals that the future solution should meet. According to [5] this should be the result of a requirements elicitation process consisting of the identification of stakeholders, of their stakes, and of the requirements derived from their motivations and stakes. TABLE 1 and TABLE 2 provide the information of the SSLEA requirements analysis, based on a study of the literature (see section III) and on interviews we carried out with representatives of companies that participated in this research, and its associated case study. The collected data was then analyzed and distilled in the form of motivation models for each separate stakeholder, an example of which can be seen in Figure *1*.

TABLE 1 Mapping of slots (prescribed by the onion model, [6]) to stakeholder types)

Slots	Roles	Stake
Normal operator	Transportation manager (LSP)	Increase responsiveness to disruptions, reduce operational risks, guarantee SLAs to customers
Operational support	System integrator	Coherent and integrated IT architecture
Maintenance operator	System administrator	Software and hardware maintenance and configuration
Functional beneficiary	Customers	On time delivery of goods, order tracking & immediate incident notifications
Regulator	Government	Implementation of/compliance to regulatory & legal measures by design, e.g., GDPR

The LSP and customer role are represented by a large Dutch fresh and frozen food LSP, delivering temperature controlled perishable products to its customers in the health care industry. The company owns a fleet of more than 100 trucks with temperature controlled trailers. The system integrator and system administrator role are represented by a renown Dutch software consultancy with various customers in the logistics industry.

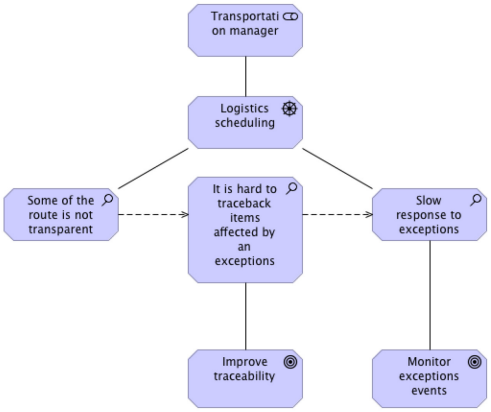

Figure 1. Motivation model of the transportation manager

TABLE 2 Elicited requirements

Requirement	Stakeholder
Agnostic data source integration	LSP, System integrator, System administrator,
Simple business processes	LSP, System integrator, System administrator
Aggregated and linked data usage	LSP, System integrator, System administrator
Nearly real-time detection	LSP, System integrator, System administrator
Information sharing	LSP (and other supply chain partners), customer, system integrator
Information security	LSP (and other supply chain partners), customer and government

Based on the stakeholder analysis, we derived a set of main requirements for the future SSLEA, presented below.

1) Agnostic data source integration, which assumes that the architecture must support a wide variety of monitoring sensors, is a key requirement. There is a broad selection of sensor technologies and functions, especially after the advent of IoT and Industry 4.0. Different sensor manufacturers might use a different communication protocol. In the future even more sensor types might become available, or a specific sensor manufacturer might cease to exist. Therefore, it is important to have the capabilities to integrate any type of new sensors, and avoid vendor lock-in. This will help the system integrator to choose the correct devices for the company. As well as the system administrator managing the lifecycle of the IoT devices.

2) A *Simple business process* asssociated with a SSLEA should not cost much effort to implement. Simplicity is required to ensure the ease of integrating the sensor technology into the existing application landscape, and acceptance in/ alignment with the business. It would benefit the system integrator and would help the transportation manager to be effecient in solving logistics exceptions.

3) Aggregated and Linked data usage makes possible the enhancement of the detected exception with other data types (e.g., location, traffic conditions, weather etc.) that would be useful in the transport manager's decision making and intervention process.

4) Nearly real-time exception detection is required to make possible *mitigation of risks* and the resilience of the *logistic service provisioning.*

5) Information sharing is required throughout the respective supply chain for improving the coordination of the cross-chain processes and the interoperability between partners.

6) Information security is needed for protecting the customer and all parties involved in the supply chain.

III. Literature Review

The next step we took in our design approach was the execution of a systematic literature survey based on [35]which as main goals to have an overview of the extant literature with regard to the following issues: 1) the reference architectures for managing data generated by IoT sensors, 2) approaches for nearly real-time exception detection, and 3) architectures for integration and sharing of data coming from multiple sources. We present a summary of the selected literature.

There are many architectural components described in the literature. For example, for information gathering, literature covers a variety of IoT protocols, such as CoAP [12], 6LoWPAN [13] and Bluetooth Low Energy [14]. The protocols focus on how the devices can be connected to the network. Much research has also been performed regarding the security of IoT [15].

With respect to nearly real-time exception detection with good accuracy, computing resources and a detection paradigm are critical. Due to the large amount of streaming data generated by sensors, storing the data using big data technology has been proposed [16]. The data can be processed within a cloud-based [17], or a fog-based architecture [18]. The main difference between the two is that in fog computing, data processing is delegated to the sensors itself or to a computer system that is responsible for managing the sensors. Examples of tasks that may be processed inside the fog include validation and data pre-processing.

With respect to the detection paradigm, the capability of handling and analyzing massive data streams coming from distributed IoT sensors is the most challenging. The complex event processing approach [19] processes data through the pipeline of a filter algorithm. In multi-agent based architectures [20], the information is processed using multiple agents each encapsulating different algorithms. Results are generated through the agents' consensus. In knowledge-based architecture [21], a knowledge database is used; both the user and some algorithm can generate the knowledge entries for the database. The system categorizes the filtered events by using suitable patterns or models from the knowledge database. When the events correlate with existing pattern or rules, the case database, containing information needed for the mitigation, is queried.

A layered architecture [22] provides the overall structure for all of the components mentioned above. It consists of the application layer, network layer, and perception layer. The perception layer is responsible for gathering data from the real world. The network layer is responsible for transmitting the data across the application, and the application layer transforms the data into digestible information. A variation of this architecture exists, in which business, management, and security layers have been added [23]. The layered architecture, however, does not provide a good solution for information sharing.

One of the prominent solutions for information sharing is Industrial Data Space (IDS) [24] architecture. The IDS is trying to solve this issue by creating a standard for the data market. IDS is an architecture to transform data into commodity. The users of the IDS can share and sell data across different industry sectors through a context mapping system called vocabulary.

The EPCIS architecture [25] also provides interesting insights concerning information sharing. The EPCIS architecture specifies standards for logistics traceability, and states that an event must be described in terms of four properties: what, when, where and why [25].The "what" identifies the type event such as "Temperature Change detected". The "where" specify the location of the event and the "why" contains the sensor data itself, for example, "10 degree Celsius". This view is primarily useful for standardizing the data structure retrieved from sensor technology, and makes the pre-processing of input data easier for various algorithms using a common format.

IV. Architecture

Based on the identified key requirements, and the state of the art literature review, we propose an architecture for situation-aware logistics. We model the architecture using the ArchiMate modeling language [26], and by means of three different ArchiMate viewpoints: the *layered viewpoint* which gives an overview of how the new IoT infrastructure can be integrated in the operational business and logistics processes, the *business process viewpoint* which zooms in the different business processes that are impacted by and use this IoT data, and the *information structure viewpoint* which depicts the detailed data architecture of the proposed solution.

A. Layered viewpoint

As it can be seen in Figure *2*, the first layer of the SSLEA is the business layer. The business layer describes functions and business processes required to realize a generic situation-aware smart logistic service. The exception management function consists of three business processes, the exception handling process, the information sharing process and the knowledge management process.

In the exception handling process, we start by gathering information necessary to decide on how to handle an exception. The information required includes what is the current situation, who is responsible, which transported items are affected and which customer is affected. If the exception is something that has happened in the past, we can retrieve the best practice to handle such an event stored in the knowledge database. This makes exception handling faster.

The application layer supports the business process by providing access to information and functionality required by the business process. It consists of four functions: absorption, processing, presentation, and security. The first three functions are derived from the 3-layer IoT architecture proposed in [27], which we supplemented with the security

function. This structure is also quite intuitive as it follows the local steps of collecting and processing streaming (big) process execution data. The absorption function is realized by two application processes: the data acquisition process, and the data validation process. The data acquisition process defines how to convert sensor specific data types from various sensor manufacturers into data that can be analyzed and interpreted by the detection function of the SSLEA. The data validation process is necessary to ensure that only valid data, especially sensor measurements, flow into the exception detection process. Invalid data might cause invalid detection which can further cause loss. The way we implement both processes is by using an enterprise service bus. An enterprise service bus acts as a message router between the various data sources and system functionality. It uses pipes and filter design patterns. Incoming messages are filtered, validated, and transformed into an appropriate format through rules stored in the knowledge base.

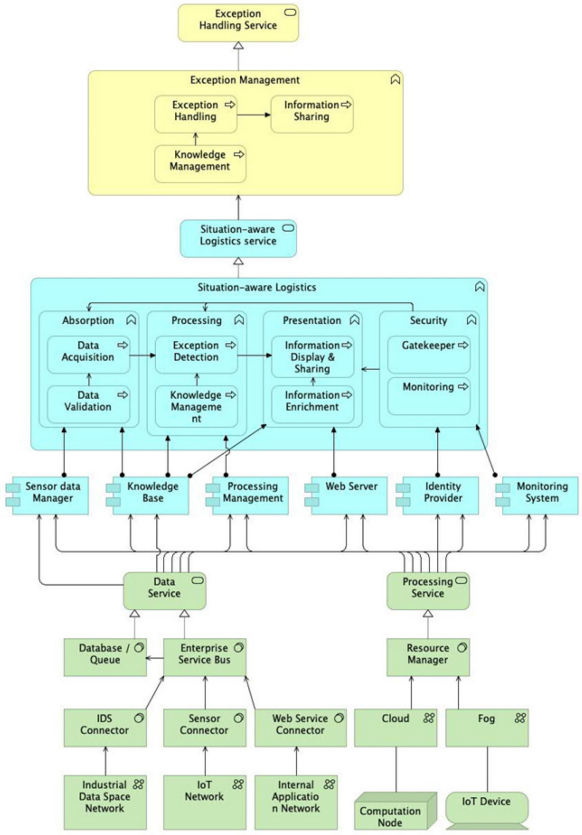

Figure 2. Layered viewpoint of the SSLEA

After the absorption process is done, the processing function of the SSLEA is triggered. At this point, two important processes take place: the exception detection process and knowledge management. A variety of approaches can be used to implement the exception detection process. This is most often rule-based, and involves a quite simple threshold-based detection mechanism. First, the data spanning a certain time interval is aggregated. For example, for temperature controlled transport, a 5-minutes average of temperature measurements could be used. This measurement is then compared with a threshold, or with a set of rules stored inside the knowledge base system. If the detection algorithm detects something unusual, it will trigger the notification function. The knowledge base can be enhanced to be more accurate in three ways. The first one is by employing user input. In this case, the user directly inputs the knowledge into the knowledge base. The second one is by using feedback: the result from the exception detection process can be fed into the knowledge base to improve it. The last one is by incorporating third party knowledge, such as information acquired from an IDS.

The presentation function takes the exception information and transforms it into information that is useful for the end-user, in the form of notifications and reports. This is presented to the user in terms of what, when, where and why the exception occurred. The information is further supplemented inside the enrichment process with linked data from other systems or third party sources. For example, if the exception concerns the engine of a certain vehicle, the system can pull information from the ERP system (first party information) regarding the vehicle's identification information, and then link it to third-party information using the industrial data space of the vehicle's manufacturer on how to perform engine diagnostics. We argue that combining exception information with accurate linked data can make exception handling faster, leading to exception cost minimization.

The last function of the system is concerned with security. It consists of two processes, gatekeeping and monitoring. The gatekeeping process ensures only an authorized user can access sensitive information such as the identity of the client. An identity provider application component supports this gatekeeping process. This application implements standard user identification and access rights enforcement protocols such as *SAML*, *Kerberos* or *LDAP*. All system functions are also monitored by a monitoring application, such as *Nagios*, to make sure the applications run as intended, preventing denial of service attacks.

In the infrastructure layer, one can see two services provided by the infrastructure: the data service, and the processing service. Both services are used by all applications included in the SSLEA. The data service is essentially an enterprise service bus, that will route and validate data from internal and external data sources (i.e., the IDS and the IoT networks), as well as the sensor network. The enterprise service bus routes the data to the application that needs it. The information is also stored in some database for storing historical data. For the internal applications, a direct connection using web services can simplify the communication.

The second infrastructure service is the processing service. It provides computing resources such as CPU time and memory. There are two ways computing can be performed: through the cloud, or the fog. The cloud means the processing capabilities are provided through the Internet. This is suitable to provide a resource for computation hungry processes such as data transformation and exception detection. The fog, which consists of computing resources in the vicinity of the physical sensor has limited computing power. Thus, the fog is suitable for rather simple data validation tasks or very simple business logic. Because the system uses the pipeline architecture, we can distribute the filter and transform algorithm to multiple computation nodes. The capability to distribute the work makes the architecture scalable. The resource manager is responsible for assigning

computational resources through a filter or data transformation algorithm.

B. Business process viewpoint

1) Exception handling process

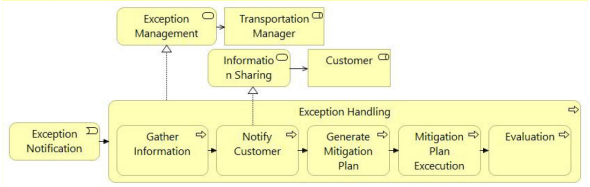

Figure 3 Exception Handling Process

The exception handling process is triggered when an exception from the IoT sensor is detected. The exception manager must determine whether the detected exception is correct or not by checking the actual situation. Once confirmed, the exception manager can start notifying the affected customers, by invoking the information sharing business process. At the same time, the person can use the information provided by the system to create an exception mitigation plan. After the plan is executed, the plan is evaluated, and the loss caused by the exception can be calculated. The evaluated plan is then stored inside the knowledge base for re-use in solving future exceptions. This is process flow is depicted in Figure 3.

2) Information sharing process

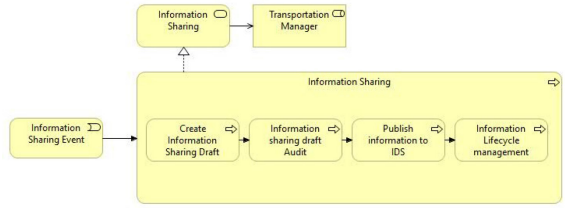

Figure 4 Information Sharing Process

Information sharing is bidirectional: not only we have to share information with an external party; we also need information from them. This process is facilitated by the IDS. The transportation manager must first perform a security audit for the information to be shared. This is important to ensure privacy. We call this process information sharing drafting. After that, the transportation manager can publish the draft through the IDS information broker. The connector and data format can be published through the IDS app store, which simplifies the integration with the client app. The final step deals with determining the lifecycle of information. Any information may be useful within a specific timeframe. This is especially true for time-sensitive information. After some time, the value of the information may decrease. Therefore, information lifecycle management is important. Unused information must be disposed of. We can also use a similar process for retrieving information from the third parties. This process is depicted in Figure 4.

3) Knowledge management process

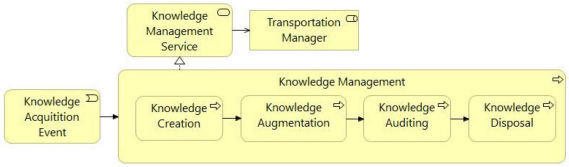

Figure 5 Knowledge Management Process

Figure 5 shows that the knowledge management process consists of four steps: knowledge creation, augmentation, auditing, and disposal. The knowledge creation process is started by the user, who is inputting rules for a certain type of exception. After that, the information is augmented by the feedback generated by the detected exception. A third party information source can also enhance knowledge. The next step is auditing. This process is necessary to ensure the knowledge in the system is accurate. Any old and unused knowledge will be marked in this process. The marked knowledge then can be disposed of. The disposal will make the detection process performant and accurate because the system only evaluates the data against an approved set of rules.

C. Data structure viewpoint

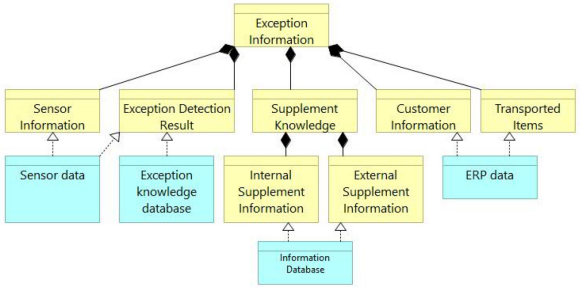

Figure 6 Data Structure Viewpoint

The most important data in this architecture is the exception information. The exception information is the main output of the SSLEA. It is used by the transportation manager to understand what happened during an exception and to notify the customer. Exception information aggregates several other types of business and data objects as shown in Figure 6. The first one is sensor information. The sensor information is in essence a measurement of some characteristic of the environment (e.g., temperature) at a certain moment in time, when the exception took place. It is being used in combination with the exception knowledge database to create the exception detection result. The supplement knowledge is additional information that is related to the exception. The source can be internal or external. An example of internal information is the driver information, whereas an example of external information is a diagnostic guide. The customer information is coming from the ERP system. This information is crucial to determine who should be notified when an exception occurs. The information regarding the items being transported is crucial. For example, the weight and the dimension of the transported items can be used to plan how many workers are needed to mitigate the exception. This is illustrated in Figure 6.

During our interview with the transportation manager, we discovered that the most important information required for mitigating exceptions is: what happened, how important it is, what items are affected, who is responsible for the exception, who is the customer, when and where did the exception take place. These requirements can be perfectly mapped onto the GS1 notification model [25], as follows: the "what" field contain the type of the detected exception , the "who" field contains the ID of the sensor, the "when" field contains when the exception happened and the "why" field contains the reason the notification is generated.

To identify events, we decided to use GS1 standards to facilitate data interoperability. Other standards we adhere to, and which can serve the same purpose, is Global Shipment Identification Number (GSIN). For example, we can use GSIN to identify the shipment, Global Trade Item Number (GTIN) to identify the items, GS1Prefix to identify the customer, Global Individual Asset Identifier (GIAI) to identify the vehicles, and Global Location Number (GLN) to represent a known location such as the production center, distribution center, and the customer address.

The system will detect an exception by using rule-based, complex event processing techniques. An item type (for example, an ice cream) is modeled in an entity called the exception category. The exception category contains multiple rules. The rules comprise of the name of the rules, threshold value, comparison operator, message, event rates, and aggregation function. The comparison operator is used to compare the value of the sensor to the threshold value. The comparison operator is a standard mathematical operator, such as less than, more than, equal, less than equal and more than equal. The system can aggregate the events, according to different mathematical functions: minimum, maximum, average, sum and count.

V. Prototype Development

In order to test and evaluate the proposed architecture we have instantiated the SSLEA by means of a prototype to support the scenario of transporting perishable goods. Although, this

A. Prototype business process

Figure 8 illustrates the transportation process that was used to test and validate the prototype. The transportation process begins when the *"order-line ready for shipment"* events raised from the ERP systems. Then, the transportation manager will schedule the order-lines for shipment, using the picking systems. The picking systems will determine the loading time, the vehicle, the goods locations inside a pallet and the pallet locations inside the vehicles. The scheduling process determine the priority of the order-lines based on the expiration date, the weight of products and the cooling requirements.

The next step is to load the order-lines into the pallets. The "loading/unloading employee" role performs the process. The employee uses the scheduling data received from the picking systems to load the order lines into pallets correctly. After that, the loading/unloading employee will load the pallets into the vehicle. Then, the vehicle transports the pallets to the distribution center. Unloading employees then unload the order-lines.

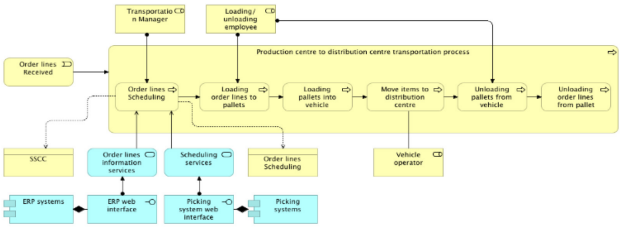

Figure 8 Transportation process

B. Prototype architecture

The prototype is designed to support the exception handling process by monitoring the condition of the goods between two locations and automatically detect exceptions. IoT sensors are used to collect environment data and push information through mobile networks to the prototype. The prototype architecture is displayed in Figure 9.

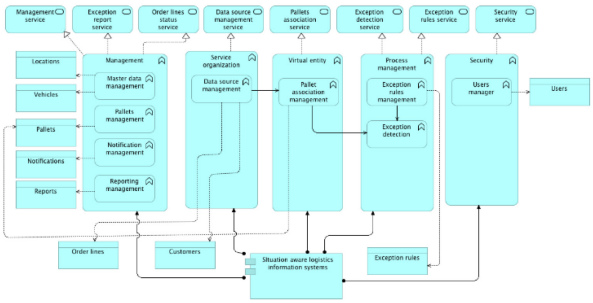

Figure 9 Prototype architecture

C. Prototype implementation

The prototype consists of an implementation of all the modules specified in the SSLEA, as well as the test unit for each module. The prototype, however, is not suitable for a production environment as it does not provide a mechanism for scaling and data durability, and as such it is a simplified instance, yet functional version, of the full SSLEA. Furthermore, the user interface is only providing essential functionality needed for illustration purposes due to development time constraints.

We used Mendix [1] as development platform for our prototype. Mendix is a model-driven development tool that makes it possible to build applications in a fast and agile manner. For the IoT devices, we connect to APIs provided by a smart pallet manufacturing company. As stated in the architecture, one of the core components is an enterprise service bus, which has been implemented using the eMagiz[2] enterprise service bus solution. eMagiz was a logical choice as it seamlessly integrates with Mendix applications, and because it has been made available by the company that provides support and the data for this research. The resulting application is hosted inside the Mendix cloud services environment. Figure 10 gives an impression the prototype's notification dashboard interface.

[1] https://www.mendix.com/

[2] https://www.emagiz.com/

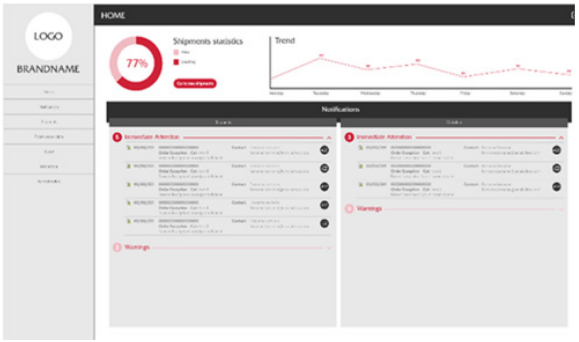

Figure 10 Notification dashboard

With respect to the IoT sensing architecture, for the prototype we used smart pallets. These are equipped with sensors that can be used to continuously monitor the state of the pallet and of the products placed on it, as will be described in more detail in the next section.

VI. Validation

A. Validation scenario

This first architecture validation is mostly concerned with a measurement of the effectiveness of the proposed SSLEA design. Further validation will follow as part of a larger research process. The EA discipline provides principles and guidelines to help organizations through the business, information, process, and technology changes necessary to execute their strategies [2]. Thus, we will use these principles to investigate two things: a) the extent to which the proposed architecture and its prototypical implementation meets the original requirements, and b) the actual performance/ability of the SSLEA's implementation to solve a given problem in a specific context.

The first validation goal can be achieved by designing a scenario (use case) that is representative for the original user requirements, ask the stakeholders and experts to perform the defined scenario and then ask their opinion about how they experienced the system.

The second validation goal (i.e., measuring performance) is linked to the primary system functions: to make logistics exception handling faster and make sure the customer is aware that something is happening as soon as possible. Since these functions are highly correlated, it is sufficient to measure the throughput time from event occurrence to customer notification.

The scenario we use for the experiment has been provided by a supplier of fresh and frozen food that has its own fleet of trucks. More precisely, the case refers to the transportation by truck of fresh and frozen food from a distribution center to a supermarket chain. The products are transported in refrigerated containers, and placed on smart pallets. For this scenario, we use standard 20 feet containers (width 235cm & height 589 cm) that can hold 11 Euro pallets (width 80 cm & height 14.4 cm). Therefore, if we assume the container is fully loaded, the shipment will contain 11 pallets. To ease the testing, we assume a shipment will contain 99 order lines. The order lines contain randomized customers. From an interview with logistics experts we learned that, usually, a container is filled with items of the same type. The reason is that each product type needs specific temperature conditions. For example, milk is transported at +6°C, while for transporting meat one needs -10°C [28]. Based on this knowledge, we assume the vehicle will transport one type of product.

In the actual experiment we chose to simulate the behavior of the refrigeration units, and generate synthetic data, as we have more control on the quality of the streaming sensor data, data pre-processing is not needed, and the experiment's set-up costs are much lower.

To simulate an exception, we need to know the characteristics of the transported item and a sensor event generation function. In our scenario the transported item is ice cream, which normally needs a temperature below -25°C [28]. The behavior of ice cream at different temperatures is discussed in detail in [29] and [30], for example that ice cream temperature would increase by 5°C in one hour in a defective insulated refrigeration unit. The latter characteristic means that if the temperature at the time of failure was -25°C, one has exactly one hour to intervene, before the temperature reaches -20°C, and the whole cargo becomes wasted.

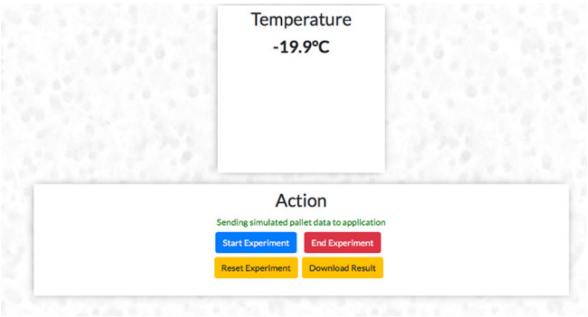

Figure 11 Temperature Sensor Simulator

To simulate the occurrence of failures we developed a sensor event generator that connects to the prototype through the enterprise service bus, and generates temperature values. The event generator can simulate the temperature increase as described above. Figure 11 shows the event generator application.

B. Validation Results

1) Fullfilment of stakeholder requirements

After seeing a demo of the prototype, the test subjects (domain experts, i.e. logistics planners) were asked to work with the system and perform activities as indicated in the validation scenario. After that, they were asked to fill in a questionnaire. For answers we used a 1 to 5 Likert scale, with 1 indicating a certain requirement is not met, and 5 that the requirement is fully implemented in the prototype. TABLE 3 contains the result of the validation.

TABLE 3 Results of expert questionnaire

Stakeholder	Requirement realization	Requirement	Opinion	Score
Transportation manager	Monitor Exception Occurrence	Agnostic data source integration Near real-time exception detection	+ Sensor gives the user more information about the condition surrounding the items + The rule-based system is extensible. The rule for the new item can be added quite easily. + The rules can be adjusted on the fly. Useful to adjusting the sensitivity. + The notifications are sorted by importance and provide necessary information to act upon - There is no well-tested rule currently available to use. - The current prototype does not have a function to mark the notification as read / handled	4

	Improve Traceability	Information augmentation Information sharing	+ The sensor provides real-time & location of an item. It may be useful for deciding mitigation operation - On the road, the system tracked the items on the pallet level. An item can be removed from the pallet on the way, untraced.	4
Customers	Timely exception notifications	Near real-time exception detection Information augmentation	+ The exception detection working as intended + The exception notification provide information about the affected customer. - No 3rd party data is currently available to enhance the detection capabilities	4
	Monitor items conditions	Information Sharing	+ The exception notification provides enough information about the affected customer. + The Industrial dataspace provides 3rd party information that can enhance the information surrounding the exception - Information security audit is a complicated process. - Information data space is hard to set up	4
The government	Adhere to security & privacy laws	Information Security	+ The role-based access limits well what the user can view. Only privileged roles can access private information such as customer information. - Complex information audit process.	2
System designer	Simple to implement & updates	Simple Business Process	+ The system module satisfies the stakeholder's need	4
System Implementer	Easy to learn	Simple Business process	+ The navigation of the user interface follows the design principles and is clear - Some of the user interfaces are too crowded with data	3
Average				3.57

The designed system scored on average 3.57. This score is above average, which actually confirms that experts consider that the prototype in its current form reflects well the intended functionality. However, it also clearly indicates that the prototype is not ready for operation as it still has a number of limitations resulting from the assumptions and simplifications imposed by this experiment.

The experts were satisfied with the detection capabilities. However, they also raised questions during the discussion. For example, they asked how the normal transportation conditions for the different types of products are established. This is important as LSPs normally deal with a large number of different products, and in the current prototype design it would be the task of the transportation manager to manually put this information in the system, entirely based on their domain expertise. This could be probably automated if product storage and transportation information would be obtained from the manufacturer, possibly by means of information sharing through an industry IDS. As such IDSs are not yet available, the experts were skeptical regarding the possibility to meet this requirement on short term. Furthermore, at the time of this experiment, we learned during an interview that the company participating in the experiment only deals with a limited number of item types, which can be handled manually by one person, which meant that this issue was not critical for this setting. As far as the issue of false alarms is concerned, the system allows the user to improve the exception rules on the fly, and the transportation manager can dynamically tune these rules to improve the precision/recall of the system. The test subjects were satisfied with this feature.

2) System performance

To evaluate performance, we measured the time taken by the test subjects to execute the exception handling process described in the architecture. We repeated the test five times per test subject and calculated the average time. There are five test subjects specialized in exception handling that performed the test.

The result is that the test subject can process the exception notification and notify the affected customers in less than 14 minutes for 49 customers. This is significantly faster, compared to almost 2 hours that is needed to do the same without using our system. The result does not include the time to manually check the current condition of the product.

The experiments produced expected and unexpected insights. On the one hand it was possible to precisely assess to what extent the handling process has been improved in terms of performance, and to conclude that the second validation goal has been achieved. On the other hand, because the prototype pushes extensive information surrounding the exception at once, test subjects suddenly were faced with information overload. At first, this led to some confusion and the performance of the human operator decreased. After the operators learned which information they should look for, the situation improved. Our original motivation to show the complete information about each event was based on the argument that for different exception types different kinds of information is needed. Thus, it is the task of the transportation manager to filter out some of the information and use only what is essential. Currently, there is no way to filter the type of information presented on the screen. This situation led to the conclusion that filtering out some of this information, by letting the operator configure the tool according to his specific needs, should be further researched.

Clearly that in the solution we proposed, exception handling involves human oversight. Furthermore, also information sharing involves expert oversight. One may argue that this will impact the ability to deliver real-time responses to logistics exceptions. However, we argue that this should not be a major problem considering that logistics processes are relatively slow, and response times of a few minutes are acceptable (even when we consider other modalities, such as air transport). Moreover, this situation is a considerable improvement compared to the current situation in which no proper (nearly) real-time monitoring and control exist of transportation processes in progress.

VII. Conclusion & Future Work

We successfully developed the architecture for a situation-aware logistics information system. The architecture consists of four parts, the business process, the application component model, the data model, and the physical model.

A. Research Results

1) What are the requirements for smart logistic planning systems to manage the risk?

We gathered the requirements based on a stakeholder analysis. We analyzed and summarized the requirements into six basic principles. Because the data about exceptions can come from a wide range of sensor manufacturers, *agnostic data source integration* is needed, which means the architecture should have the capabilities to ingest diverse data sources. To minimize the effort to implement changes in the organization, a *simple business process* is needed. To be able to enhance the information about the exception with related data, useful to solve the exception, *information augmentation* should be supported. To allow the transportation manager to timely react, *near real-time exception detection* must be supported. *Information sharing* is useful for sharing exception

information to the customer or another stakeholder, as well as for acquiring knowledge from the third party. Finally, *information security* is important to ensure stakeholders security and government rules.

2) What is the architecture of situation-aware logistics system?

We based the architecture on existing architectures described in the literature, with addition and removal of components based on the requirements. The main reference architecture that we use is the layered event processing architecture. The reason for this is that the layered architecture provides most of the components, tools, and methods that the requirements specified. Since the layered architecture lacks a solution for data sharing, we use the data model for describing sensor events from the EPCIS architecture.

To provide communication with external data sources, we use an enterprise service bus along with the Industrial Data Space architecture. We use an enterprise service bus to adapt, route and validate various data format from all of the data sources that might use various communication protocol. Additionally, the enterprise service bus also provides tools to secure communication between the external systems.

3) What is the performance of the architecture in reducing transportation risks?

We have defined the performance based on the goal of the information system. We formulated the following two hypotheses: (1) by using the system, the stakeholders' requirements will be fulfilled; and (2) by using the system, exceptions can be detected faster, leading to timely customer awareness. To validate the hypotheses, we have created a prototype and tested it by feeding synthetic test data.

For the first hypothesis, the answer is yes; the system can fulfill the stakeholder requirements. However, some improvements are possible. For example, currently, there is no backup solution and method to retrieve historical sensor data in the architecture, because we use a pipeline architecture to process the information. In our defense, the processed information is more useful than raw sensor data. We argue that sensor data value is depreciated as time passes by. Therefore, if we want to save historical data, we should save only the processed information.

For the second hypothesis, the test subjects produced the notification to the customer faster than without the system. However, the user questions whether the system is useful when the exception happens near to the destination, as the value of mitigating the exception diminished, because the vehicle can move forward. In our opinion, having the exception information is still useful, to study the cause of the exception and to prevent future exceptions to happen.

B. Limitations

We tested the architecture using synthetic data. The test result may not reflect how the architecture will behave under the realistic conditions. However, we made sure the data was as realistic as possible. The master data is based on anonymized real data. For example, the number of pallets and order lines that had to be handled was based on the dimension of a real container. We generated the event based on a real model of temperature drop inside an insulated container and a real model of perishable items.

The last identified limitation is the scope of the validation. We focus on the validation of the architecture performance. There are many aspects of the architecture that still needs validation — for example, validating the assumption that the architecture is secure, scalable and adaptable.

C. Contribution

Prior research has shown us many architectures for leveraging the IoT into the logistics fields. However, the main purpose of these architectures is to identify and record the location of transported objects, which is vital for traceability. One survey [22] listed the relevant architectures. One example of a relevant architecture is EPCIS [25], which is a GS1 standard. However, there are not many architectures available for detecting logistics exceptions and providing useful information to react. Another example is EURIDICE [4], which supports the gathering of valuable information for logistics operations. Nevertheless, none of the mentioned architectures focuses on the exception handling process. This study attempts to fill this gap by defining the roles and presenting the required change to the organization for exception handling. In this study, we have developed an architecture to serve as guidance for implementing the concept of situation-aware logistics.

D. Future Work

The next logical step is to research how to incorporate artificial intelligence into the architecture. Artificial intelligence could play a role to tune the exception rules, so that more accurate notifications can be produced, and to perform postmortem analysis. There are many questions for integrating artificial intelligence into the architecture. What kinds of AI algorithms are suitable? What are the requirements on the input data? How should the results of the AI algorithm be presented?

We also plan more validation experiments to test the architecture. Currently, the architecture is only validated with synthetic data. It is expected that thoroughly testing the architecture in a real life situation will provide new useful insights for improving the architecture.

Finally, we want to improve the functionality and usability of the architecture. Several questions need to be answered as part of this objective. What improvements are possible for the architecture in order to better contribute to stakeholder goals? How do these improvements extend the functionality and/or increase the performance? What is the usability and usefulness of the improved architecture?

REFERENCES

[1] M. Klein and C. Dellarocas, "A Knowledge-based Approach to Handling Exceptions in Workflow Systems," *Comput. Support. Coop. Work*, 2000.

[2] B. Cameron and N. Malik, "A Common Perspective on Enterprise Architecture," The Federation of Enterprise Architecture Professional Organizations (FEAPO), pp. 1-12, 2013.

[3] B. Li, R. Yang, and Y. Hu, "An Experimental Study for Intelligent Logistics: A Middleware Approach," *Chinese J. Electron.*, vol. 25, no. 3, pp. 561–569, 2016.

[4] M. Forcolin, E. Fracasso, F. Tumanischvili, and P. Lupieri, "EURIDICE - IoT applied to logistics using the Intelligent Cargo concept," in *Concurrent Enterprising (ICE), 2011 17th International Conference on*, 2011, pp. 1–9.

[5] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[6] I. F. Alexander, "A Taxonomy of Stakeholders," *Int. J. Technol. Hum. Interact.*, 2005.

[7] J. Grabara, M. Kolcun, and S. Kot, "The role of information

systems in transport logistics," *Int. J. Educ. Res.*, 2014.
[8] T. P. Stank and T. J. Goldsby, "A framework for transportation decision making in an integrated supply chain," *Supply Chain Management*. 2000.
[9] S. M. Wagner and C. Bode, "AN EMPIRICAL EXAMINATION OF SUPPLY CHAIN PERFORMANCE ALONG SEVERAL DIMENSIONS OF RISK," *J. Bus. Logist.*, 2008.
[10] C. Tankard, "What the GDPR means for businesses," *Netw. Secur.*, 2016.
[11] ISO/IEC, "ISO/IEC 27000 Information technology - Security techniques - Information security management systems - Overview and vocabulary," *ISO/IEC*, 2014.
[12] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," 2014.
[13] G. Mulligan, "The 6LoWPAN architecture," 2010.
[14] A. F. H. Iii, V. Khanna, G. Tuncay, R. Want, and R. Kravets, "Bluetooth Low Energy in Dense IoT Environments," *IEEE Commun. Mag.*, 2016.
[15] A. Riahi Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A Roadmap for Security Challenges in Internet of Things," *Digit. Commun. Networks*, 2017.
[16] M. Marjani *et al.*, "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[17] A. Jede and F. Teuteberg, "Towards cloud-based supply chain processes: Designing a reference model and elements of a research agenda," *Int. J. Logist. Manag.*, vol. 27, no. 2, pp. 438–462, 2016.
[18] T. S. J. Darwish and K. A. Bakar, "Fog Based Intelligent Transportation Big Data Analytics in The Internet of Vehicles Environment: Motivations, Architecture, Challenges and Critical Issues," *IEEE Access*, 2018.
[19] V. Akila, V. Govindasamy, and S. Sandosh, "Complex event processing over uncertain events: Techniques, challenges, and future directions," in *2016 International Conference on Computation of Power, Energy, Information and Communication, ICCPEIC 2016*, 2016, pp. 204–221.
[20] N. Jabeur, T. Al-Belushi, M. Mbarki, and H. Gharrad, "Toward Leveraging Smart Logistics Collaboration with a Multi-Agent System Based Solution," in *Procedia Computer Science*, 2017, vol. 109, pp. 672–679.
[21] D. Xu, C. Wijesooriya, Y. G. Wang, and G. Beydoun, "Outbound logistics exception monitoring: A multi-perspective ontologies' approach with intelligent agents," *Expert Syst. Appl.*, 2011.
[22] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University - Computer and Information Sciences*, 2016.
[23] C. Wunck and S. Baumann, "Towards a process reference model for the information value chain in IoT applications," in *2017 IEEE European Technology and Engineering Management Summit (E-TEMS)*, 2017, pp. 1–6.
[24] B. Otto, M. ten Hompel, and S. Wrobel, "Industrial Data Space," in *Digitalisierung*, 2017.
[25] GS1, "EPC Information Services (EPCIS) Standard," pp. 1–138, 2016.
[26] M.-E. Iacob, H. Jonkers, M. M. Lankhorst, H. A. Proper, and D. A. C. Quartel, "ArchiMate 2.0 Specification," *Arch. 2.0 Specif.*, 2012.
[27] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial Internet: A Survey on the Enabling Technologies, Applications, and Challenges," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1504–1526, 2017.
[28] S. A. Tassou, G. De-Lille, and Y. T. Ge, "Food transport refrigeration - Approaches to reduce energy consumption and environmental impacts of road transport," *Appl. Therm. Eng.*, 2009.
[29] M. R. Muse and R. W. Hartel, "Ice Cream Structural Elements that Affect Melting Rate and Hardness," *J. Dairy Sci.*, 2010.
[30] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University - Computer and Information Sciences*, Elsevier, 08-Oct-2016.
[31] H. Shah and M. El Kourdi, "Frameworks for Enterprise Architecture," *IT Professional*, 9(5): 36-41, 2007.
[32] R.K.M. Veneberg, M. Iacob, M. van Sinderen, and L. Bodenstaff, "Relating Business Intelligence to Enterprise Architecture Intelligence: A Method for Combining Operational Data with Architectural Metadata," *Int. J. Cooperative Inf. Syst*. 25(2): 1-36, 2016.
[33] M. Iacob, L. Meertens, H. Jonkers, D. Quartel, L.J.M. Nieuwenhuis, and M. van Sinderen, "From Enterprise Architecture to Business Models and Back," *Software and Systems Modeling* 13(3): 1059-1083, 2014.
[34] W. Engelsman, D. Quartel, H. Jonkers, and M. van Sinderen, "Extending Enterprise Architecture Modelling with Business Goals and Requirements," *Enterprise IS* 5(1): 9-36, 2011.
[35] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," IEEE Internet Things, no. 1, p. 86, 2015.
[36] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele Univ., 2004.