Integration of a Food Distribution Routing Optimization Software with an Enterprise Resource Planner

Pedro J. S. Cardoso[1,2], Gabriela Schütz[1,3], Jorge Semião[1,5], Jânio Monteiro[1,5], João Rodrigues[1,2], Andriy Mazayev[4], Emanuel Ey[1], Micael Viegas[1], Carlos Neves[6] and Sérgio Anastácio[6]

[1]*Instituto Superior de Engenharia, University of the Algarve, Faro, Portugal*

[2]*LARSys, University of the Algarve, Faro, Portugal*

[3]*CEOT, University of the Algarve, Faro, Portugal*

[4]*Departamento de Engenharia Eletrónica e Informática, University of the Algarve, Faro, Portugal*

[5]*INESC-ID, Rua Alves Redol, 9, Lisbon, Portugal*

[6]*X4DEV, Business Solutions, Loulé, Portugal*

{*pcardoso, gschutz, jsemiao, jmmontei, jrodrig, amazayev, eevieira, mimviegas*}*@ualg.pt,*
{*carlos.neves, sergio.anastacio*}*@x4dev.pt*

Keywords: Enterprise Resource Planning, Vehicle Routing Problem, Geographical Information, Application Programming Interface, Data Acquisition.

Abstract: *i3FR* is a system for optimizing the distribution of fresh goods, using a fleet of vehicles, which must be integrated with an existing Enterprise Resource Planning (ERP) software, while avoiding disruption of established schedules and procedures. In terms of delivery optimization, the most similar problem in literature is the well-established and experimented Vehicle Routing Problem with Time Windows (VRPTW), which is, by nature, a multiple objective problem where the number of vehicles and the total traveled distance must be minimized. In this paper, the overall architecture of the *i3FR* system is presented, which includes several modules and services (such as ERP, routing optimization and data acquisition modules), and the communication between them. The novelty in the proposed architecture comprises: (i) the application to a fresh goods distribution network, with multiple restrictions on goods' acclimatization; (ii) a data acquisition system which includes vehicle data and environmental data from the vehicles' refrigerated volumes, such as temperature; and (iii) the integration of the VRPTW optimization system with an existing ERP. A case-study is described and validates the overall implementation. The main contribution of paper is the proposal of a multi-layered architecture to integrate existing ERP software with a route optimization and a temperature data acquisition module.

1 INTRODUCTION

i3FR (Intelligent Fresh Food Fleet Router) is an ongoing project of the University of the Algarve and X4DEV Business Solutions, with the main objective of building a system to manage and to optimize the distribution of fresh products by private fleets. The system will be integrated with an existing ERP, namely the SAGE ERP X3 (SAGE, 2014), and will compute routes from the depots to the delivery points using cartographic information and taking into consideration multiple objectives.

The intelligent management of the distribution fleet is assured not only by optimizing costs, but also by monitoring what is being distributed. In the case of refrigerated vehicle fleets, for the transportation of fresh and frozen goods, the specificity is high. In this particular case it is necessary to maintain the quality of the products, to satisfy the legal criteria for food transportation, as well as the goods' safety and hygiene as required by suppliers and customers.

The product being developed can be divided in three main components: (1) a data acquisition system for the distribution vehicles; (2) an intelligent routing optimization platform, and (3) an Application Program Interface (API) to integrate the new modules with the existing ERP.

The first component will acquire statistical data from the vehicles (e.g., fuel consumption and speeds obtained from "control units"), from the past/real delivery tours (e.g., using GPS data or track obstructions reported by drivers), along with data from wire-

S. Cardoso P., Schütz G., Semião J., Monteiro J., Rodrigues J., Mazayev A., Ey Vieira E., Viegas M., Neves C. and Anastácio S..
Integration of a Food Distribution Routing Optimization Software with an Enterprise Resource Planner.
DOI: 10.5220/0005385901530161
In *Proceedings of the 1st International Conference on Geographical Information Systems Theory, Applications and Management* (GISTAM-2015), pages 153-161
ISBN: 978-989-758-099-4

less sensors placed inside and outside the transportation chambers. In the case-study, the vehicles' transportation chambers are divided into three subtypes, namely: frozen, chilled and ambient/dry goods categories. The acquired data is then used in the second component, which optimizes the routes, taking into account multiple constraints (e.g., vehicle capacities, time windows for deliveries, route duration, balance between route duration and driver working periods) and objectives (e.g., minimize the number of routes, minimize the total distance, minimize the total time to perform the deliveries and maximize customers satisfaction). The route computation will be endowed with intelligence based on data acquired during previous routes, leaving open the possibility of integrating other sources. The third module is fundamental as a bridge between the existing ERP and what is proposed to be conducted in the *i3FR* project.

Some related works and products exist on the market. An algorithm for the distribution of fresh vegetables in which the perishability represents a critical factor was developed in (Osvald and Stirn, 2008). The problem was formulated as a VRPTW with time-dependent travel-times (VRPTWTD), where the travel-times between two locations depend on both the distance and the time of the day. The problem was solved using a heuristic approach based on the Tabu Search (Glover and Laguna, 1999). The performance of the algorithm was verified using modified Solomon's problems. A somehow similar work was proposed in (Tarantilis and Kiranoudis, 2002), which deals with distribution problem formulated as an open multi-depot vehicle routing problem (OMD-VRP) encountered by a fresh meat distributor. To solve the problem, a stochastic search meta-heuristic algorithm, termed as the list-based threshold accepting algorithm, was proposed. In (Ambrosino and Sciomachen, 2007) was considered a generalization of the asymmetric capacitated vehicle routing problem with split delivery. The solution determines the distribution plan of two types of products, namely: fresh/dry and frozen food. The problem was solved using a mixed-integer programming model for the problem, followed by a two-step heuristic procedure. In (Abousaeidi et al., 2011) the distribution of fresh vegetables was addressed. The focus of the study was the delivery of fresh vegetables selecting the best routes particularly for urban areas such as Kuala Lumpur city, which faces traffic problems. There are also a relatively large number of companies providing commercial software which is similar to the one developed in the *i3FR* project (Routyn, 2014; Optimoroute.com, 2014; optrak.com, 2014; Newronia.com, 2014; Logvrp.com, 2014).

The main contribution of this paper is the proposal of a multi-layered architecture to integrate existing ERPs with a route optimization and a temperature data acquisition module. The optimization module is prepared to deal with dynamic scenarios, where new demands may appear during the optimization process, which will integrate them in the iteration procedure. Furthermore, computed routes will have several states, e.g., (1) "open" meaning that new orders can be added, removed or swapped inside that route, (2) "locked" in which case new orders may still be inserted but swapping is no longer allowed, and (3) "closed" meaning that no more changes can be made to the route. Furthermore, the system is prepared to acquire geographical data from several sources, namely from Google Maps, from an Open Street Maps router and from data retrieved from previous deliveries/routes. The system is also prepared to accept several optimizers, if necessary. All this maintaining the company's core ERP software.

A distribution company of fresh, frozen, ultra-frozen and dried products was selected as case-study, having up to 5000 daily deliveries and a fleet of 120 vehicles. Therefore, the system must be able to manage thousands of customers, each with different demand characteristics, matching the company's delivery resources with the customer's delivery requirements. The use of the company's ERP allows for maintaining most of the existing procedures, which avoids disruption to the company's routines.

The remaining document is structured as follows. Section 2 presents the steps necessary to integrate the existing ERP with the new modules, namely the Optimization module (*i3FR-Opt*), the Hub module (*i3FR-Hub*), the Database module (*i3FR-DB*) and the Maps module (*i3FR-Maps*). The data acquisition module is addressed in Sec. 3. The fourth and final section presents some conclusions and future works.

2 STEPS IN THE INTEGRATION OF AN ERP WITH A VRP OPTIMIZATION MODULE

A typical enterprise has many departments or business units (e.g., sales, inventory, finance, and human resources departments), that have to be continuously communicating and exchanging data with each other. For instance, whenever the sales department makes a sale, it has to check the inventory department and send information relative to the sale to the finance department. In the middle is the human resources department, which will be informed to ensure man power to

process the picking and sending of the items. Therefore, the success and efficiency of the organization lies, although not exclusively, in the successful communication between those departments.

A decentralized system would maintain data spread through the local departments. The local departments, in general, do not have access to the data of the other departments, which turns obtaining real data (e.g., stock inventory) into a time-consuming procedure, leading to inefficiency, loss of revenues and possible customer dissatisfaction. Simultaneously, the lack of communication makes it hard to do integrated business intelligence procedures, due to disparate, redundant and, most probably, inconsistent enterprise data.

In a centralized system, often simply called ERP, data is maintained at a central location and shared with other departments in real time. Sensitive data sharing is supported on views to the various users and departments. This strategy provides more accurate data by removing redundancy and real time updates.

In our case, the ERP management system is a SAGE ERP X3 (SAGE, 2014), which stores all vital information of the corporation activities, like customers, orders, suppliers, vehicles, employees, etc. The corporation management is done using interfaces/forms properly developed for the retrieving and filling of data stored on the ERP database.

The *i3FR* Routing System, used to compute the distribution routes, was designed to be independent of the ERP, allowing future integration with other ERP systems or even autonomous operation, given the implementation of the proper interfaces. Under that independence perspective, the communications between the systems were implemented as web services. More precisely, a Web API with simple representational state transfer (REST) based communications was considered (Richardson and Ruby, 2008). The advantage is in the fact that RESTful APIs do not require XML-based Web service protocols to support their interfaces, which in the present case was done using JSON (JSON, 2014).

The web services on the ERP side, provide the necessary data through GET requests/methods (e.g., customers addresses, orders, available vehicles, and vehicles capacities) and receive the results generated by the *i3FR* Routing System through POST methods (e.g., computed routes).

The *i3FR* Routing System is composed of several interdependent sub-modules which also communicate via HTTP, thus allowing for the system to be scaled from a single-machine environment to a large multi-server production deployment. In more detail, the system is composed of the following modules:

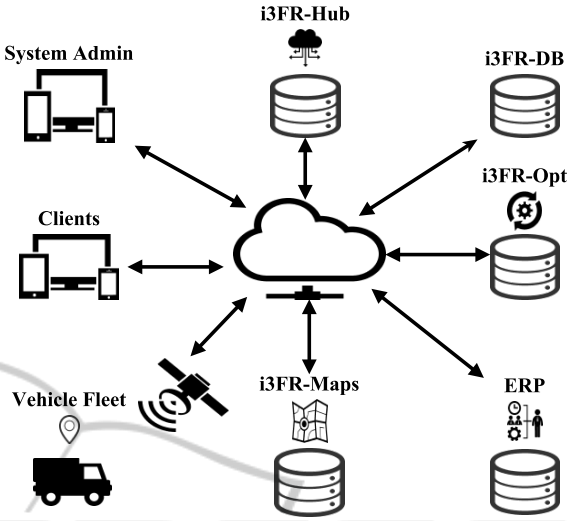

Figure 1: System's global diagram.

i3FR-Opt – The actual route optimization algorithm is implemented in the optimization module designated *i3FR-Opt*. The optimization process relies in heuristics and meta-heuristics (Cardoso et al., 2015) that should run in real-time, taking into consideration the dynamics of the overall system (more details in Sec. 2.1).

i3FR-Hub – All communications occurring inside the *i3FR* Routing System go through *i3FR-Hub* which acts, as the name suggests, as a hub. For instance, the optimization module, *i3FR-Opt*, sends and receives all the data it requires through a RESTful API provided by the *i3FR-Hub*. Furthermore, all communication to the external systems also goes through this hub, which acts as a "gateway", a security access provider, and a data validator (more details in Sec. 2.2).

i3FR-DB – The *i3FR* Routing System operates on top of a non-relational database. The database provides local storage of information relevant to the optimization procedure, namely: data retrieved from the ERP (avoiding overloading the system with constant requests), cartographic information, computed routes, etc. (see Sec. 2.3).

i3FR-Maps – A cartography subsystem, *i3FR-Maps*, was also implemented. The system retrieves routing informations from other systems (e.g., Google Maps) and stores it on the *i3FR-DB* (see Sec. 2.4).

The overall system is depicted in Fig. 1.

2.1 *i3FR-Opt* Module

i3FR-Opt module is the optimization unit, responsible for finding a suitable set of routes for the distri-

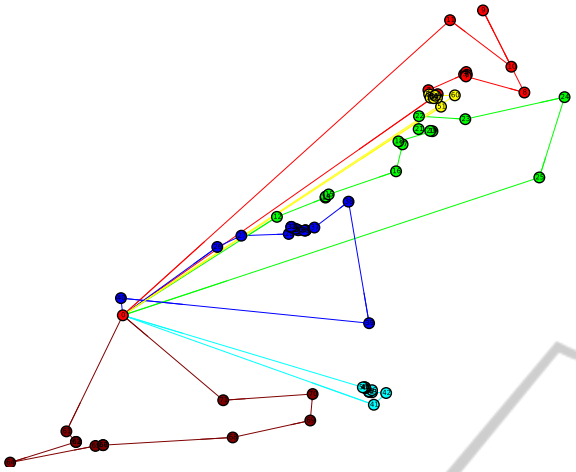

Figure 2: Excerpt of a solution returned by the *i3FR-Opt*, presenting 6 routes with 71 deliveries.

bution of fresh goods. The optimization procedure computes routes from one or more depots that visit each customer once, within given time intervals, without violating the vehicle capacities, and other legal constraints (e.g., maximum consecutive driving time). The most similar problem to the one to be solved is the Vehicle Routing Problem with Time Windows (VRPTW), common to the majority of the distribution fleets (Hsu et al., 2007; Chen et al., 2009; Faulin, 2003; Solomon, 1987; Cardoso et al., 2015). In its simplified form, the VRPTW can be stated as the problem of designing an optimum set of routes that deliver a set of goods, to a set of customers, within predefined time windows.

Within the VRPTW perspective, for the time being, the main objectives of the computational process are to obtain solutions which minimize the number of necessary delivery vehicles/drivers, minimize the total delivery distance, and maximize the minimum load between all vehicles. These objectives optimize the operational costs (e.g., fuel and worker) and give more equilibrated routes, in terms of man workload. Figure 2 depicts an excerpt of a solution returned by the *i3FR-Opt*, presenting 6 routes with 71 deliveries.

The module itself can be seen as a composition of two sub-modules (see Fig. 3). The *server* sub-module is responsible for the communication with the external services, in particular with the *i3FR-Hub* module. This sub-module acts as a server, in the sense that it waits for notifications from the hub. Whenever a notification arrives, the server sub-module gets the corresponding data, which includes the tasks of checking if the received data is consistent and sends it to the optimizer sub-module. This strategy allows to continuously maintain the optimization sub-module iterative process, avoiding blocking commu-

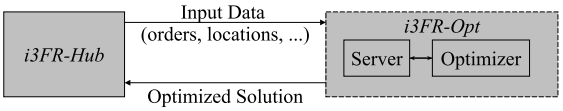

Figure 3: Communication between Hub and Opt modules.

nication. Those notifications (e.g., new/edited orders) are passed through shared memory to the optimizer, which in turn computes the best possible routes.

In the other direction, whenever significant improvements to the solution are obtained, they are communicated to the server, which posts them to the *i3FR-Hub*, and which in turn sends them to the company's ERP to be visualized and possibly adopted as a solution. Figure 4 shows an excerpt of a particular route, as a JSON document, return by the optimization module to the hub. Among other things, the route document has information about the start and end times, total distance, orders and path identifiers. The document is then expanded in the *i3FR-Hub* by dereferencing the identifiers, and the result is stored on the *i3FR-DB* and sent to the ERP.

At this point, since it is infeasible for the company to process all routes simultaneously, a method for locking/constraining optimization of certain routes was devised. As vehicles are loaded in inverse delivery order, this locking state prohibits the optimization module from changing the delivery sequence but allows other changes as the insertion of new deliveries along the computed path. This allows the company to lock a route to start picking goods and loading the corresponding vehicle. This locking command, as all communications, is relayed through the *i3FR-Hub*. This led to the need to implement a rollback procedure, such that the system could recover to the point where the locked solution was sent to the decision maker at the Interface and Management System (i.e., the ERP).

As a final note, the optimization module relies on heavy computation and was developed in C++.

2.2 *i3FR-Hub* Module

As already introduced, *i3FR-Hub* is the routing system's central communications hub, which fuses data from several sources to be pre-processed and forwarded, for instance to the *i3FR-Opt* module. *i3FR-Hub* accesses the RESTful-API provided by the Interface and Management System (the ERP) to obtain data on depots, customers' locations, vehicles, customer orders and product details.

Vehicle data includes information about the maximum (legal) transportation mass and volume capacities. In the case-study, it was considered that the vehi-

```
{
    "idVehicle": "v200_0",
    "routeStart": "2014-11-28 07:46:00",
    "routeEnd": "2014-11-28 15:39:00",
    "Distance": 61940,
    "routeState": "closed",
    "route" :
    [
        {
            "idDepot" : ObjectId("..."),
            "estimatedDepartureTime" : "2014-11-28
                07:46"
        },
        {
            "idPath" : ObjectId("..."),
            "pathLength" : 24203,
            "travelTime" : 49
        },
        {
            "idOrder" : ObjectId("..."),
            "idLocation" : ObjectId("..."),
            "estimatedArrivalTime" : "2014-11-28 08
                :35"
        },
        {
            "idPath" : ObjectId("..."),
            "pathLength" : 916,
            "travelTime" : 2
        },
        ...
        {
            "idDepot" : ObjectId("..."),
            "estimatedArrivalTime" : "2014-11-28 15
                :39"
        }
    ]
}
```


Figure 4: Excerpt of route generate by the *i3FR-Opt*.

cles have multiple transportation categories, namely: frozen, chilled and ambient/dry goods categories.

Product information includes mass per unit, transportation category (e.g., temperature range) as well as individual product dimensions, from which package volumes are computed.

Customer orders are pre-processed/aggregated before being sent to *i3FR-Opt*, i.e, the *i3FR-Hub* uses the product information to aggregate products by transportation categories. This way, the *i3FR-Opt* receives the totals for each transportation category (namely, mass and volume), allowing *i3FR-Opt* to focus on improving the solutions.

From the list of customer locations and depots, a distance matrix composed of several layers is generated (see Sec. 2.4) and stored on *i3FR-DB*. An excerpt of this data is passed to *i3FR-Opt* where it is used for computing the routes. A different excerpt, containing

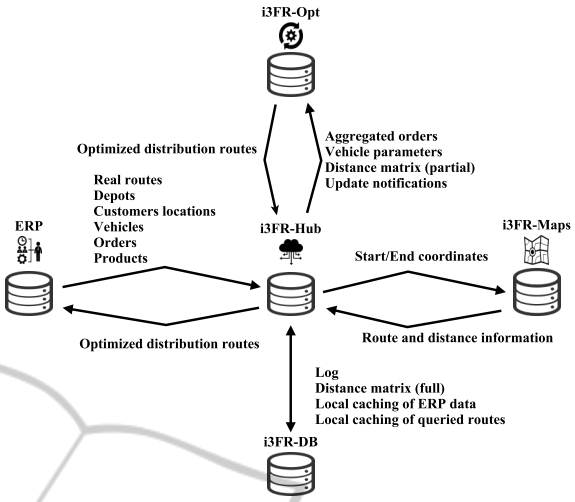

Figure 5: Inputs and outputs of *i3FR-Hub*.

full human-readable path descriptions is later passed to the Interface and Management System allowing the representation of the computed routes on the management interface and later use on the vehicle's navigation systems (see Fig 6).

A full log of operations is maintained by *i3FR-Hub*, which allows storing intermediate solutions from the beginning to the end of the optimization process, as well as rolling back and resuming optimization from previous states.

Finally, *i3FR-Hub* also provides a RESTful API for an in-development web-based back office user interface. An overview of the data going in and out of *i3FR-Hub* is shown in Fig. 5.

2.3 *i3FR-DB* Module

The *i3FR* routing system stores all data in a network connected high performance database which relies on a MongoDB server, with an ODM (Object-Document Mapper) layer implemented in Python for convenience.

MongoDB is a non-relational database, also known as a NoSQL database (MongoDB, 2014; Redmond and Wilson, 2012). This is a document-oriented database with high performance and high reliability. Other characteristics include easy scalability (vertically and horizontally through replication and auto-sharding, respectively) and map-reduce.

A MongoDB database is structured as a set of collections, which store documents. These documents are BSON objects (a binary JSON (JSON, 2014) document format), allowed to have a dynamic schema, i.e., documents in the same collection are not forced to have the same structure. This schema-less property is particularly important in the present problem, since

```
{
    "_id" : ObjectId("..."),
    "distance" : 389,
    "travelTime" : 58,
    "source" : "gmaps_directions",
    "steps" : {
        "end_address" : "Human-readable address",
        "start_address" : "Human-readable address"
            ,
        "steps" : [
            {
                "html_instructions" : "Human-
                    readable driving instructions"
                    ,
                "distance" : 319,
                "duration" : 47,
                "polyline" : "mr|aF|rnl ... @Dp@@Z
                    ",
                "start_location" : [#.#, #.#],
                "end_location" :  [#.#, #.#],
            },
            {
                "maneuver" : "turn-right",
                "html_instructions" : "Human-
                    readable driving instructions"
                    ,
                "distance" : 70,
                "duration" : 11,
                "polyline" : "wo|aFdio ... Q?U@YB[
                    B",
                "start_location" : [#.#, #.#],
                "end_location" :  [#.#, #.#],
            }
        ],
        "start_location" : [#.#, #.#],
        "end_location" :  [#.#, #.#],
    }
}
```

Figure 6: Example of an expanded path.

some of the data stored in the database does not follow a common design (e.g., data retrieved from the cartographic system). Other important factors led us to use MongoDB: (1) the possibility to embed in a single document a set of data (e.g., a solution of the problem or an order with multiple products), avoiding intricate and expensive cross-table join procedures; (2) the geospatial engine capable of geographic storage and geographic search features. Many of these features are supported on 2D and 2Dsphere indexes. The first one calculates geometries over a flat surface while the second one does it over an Earth-like sphere. In this case, location data is stored in GeoJSON objects with longitude and latitude. The engine supports fundamental operations like "inclusion" to query for locations contained entirely within a specified polygon, "intersection" to query for locations that intersect with a specified geometry, and "proximity" to query for the points nearest to another point. The last one is used for instance to set the priorities of the access to the cartographic API (see Sec. 2.4). Finally, (3) JSON was defined as the language used to communicate between all modules, which simplifies storing and generating documents, avoiding costly conversions to other data representations.

The decision of using MongoDB combines the high performance of the document storage database with the convenience of using document-like instances. At the same time, overloading the existing Interface and Management System infrastructure with high volumes of database queries is avoided, because a local cache of all relevant ERP data is maintained in the MongoDB database, along with a full operations and error log.

Figure 7 presents two examples of the structure of the JSON documents stored on the MongoDB database, namely a delivery location (on the top) and the details of an order (on the bottom).

2.4 *i3FR-Maps* Module

The database stores a multi-layered distance matrix built with data obtained from several mapping sources which contains full road information at the highest layer.

The first (lowest) layer is built using the geospatial MongoDB features to compute a complete distance matrix of geographic distances over a sphere with approximately the Earth's curvature. This is easily computed from the geographical coordinates, providing a first approximation to the real distance. However, these values are only for newer customers since more exact values will be computed later using cartographic services.

An asynchronous process then makes use of this first layer to incrementally build a second layer from a locally maintained mapping system based on a OpenStreetMaps router (OSRM, 2014). This process will work towards generating a complete matrix of queried routes, prioritizing its queries by nearest geographic distance between locations, which are more probable of belonging to a route.

Initially, this 2nd layer of the distance matrix is provided to *i3FR-Opt* to generate initial delivery routing approximations. These initial approximations are then analyzed in *i3FR-Hub* in order to gradually build a 3rd layer of the distance matrix by selectively querying an up-to-date commercial routing service such as GoogleMaps, Bing or similar.

Further iterations of the routing optimization are then based on the 3rd and most precise layer of the distance matrix.

```
{
    "_id": ObjectId("..."),
    "idErp": #,
    "loc": {
        "type" : "Point",
        "coordinates" : [ #.#, #.#]
    }
    "updateStamp": "2014-11-01 11:59:00"
}
```

```
{
    "_id": ObjectId("..."),
    "idOrder": #,
    "depot": {
            "id" : ObjectId("..."),
            "idErp" : "Unique ID as provided by
                ERP",
            "loc" : {
                    "type" : "Point",
                    "coordinates" : [#.#, #.#]
            },
            "address" : "Human-readable address",
    },
    "location": {
            "id" : ObjectId("..."),
            "idErp" : "Unique ID as provided by
                ERP",
            "loc" : {
                    "type" : "Point",
                    "coordinates" : [#.#, #.#]
            },
            "address" : "Human-readable address",
    },
    "items": [
            {
            "idProduct": ObjectId("..."),
            "storageCategory": "_Freeze_",
            "mass" : #,
            "x" : #, "y" : #, "z" : #,
            "qty": #.#
            },
            ...
    ],
    "serviceTime" : 5,
    "deliveryWindow": [
        {
            "start": "2014-11-28 09:00:00",
            "end": "2014-11-28 11:00:00"
        },
        {
            "start": "2014-11-28 14:00:00",
            "end": "2014-11-28 17:00:00"
        }
    ],
    "updateStamp": "2014-11-27 17:41:00"
}
```

Figure 7: Examples of the structure of the JSON documents stored on MongoDB: a delivery location (on the top) and an order (on the bottom).

The higher layers of the distance matrix include information on distance, travel time and toll information for multiple alternative routes between locations. Full human-readable point by point route information is also maintained which can be displayed to drivers on the in-vehicle user interface.

3 DATA ACQUISITION MODULE

The *i3FR* project also implements a data acquisition module from the vehicles included in a more general module. The acquired data will be used to validate the routes and provide information about other parameters, such as the vehicles chambers' temperatures, vehicles' fuel cost, traveled kilometers, and vehicle's Global Positioning System (GPS) data.

The overall system is based on a main controller module, powered by the vehicle's battery, which is based on an ARM microcontroller unit core and is responsible to gather information from the external sensors and devices, and to communicate with the main ERP system through a Global System for Mobile Communications (GSM) protocol. Hence, the main controller of the data acquisition system is connected inside the vehicle to several devices, as follows: (i) to the vehicle's electronic control unit (ECU), using the OBD-II standard; (ii) to an external GPS module, using a serial interface; (iii) to temperature sensors placed in the refrigerated chambers, using the ZigBee protocol (based on the IEEE 802.15.4 standard); and (iv) to an external GPRS (General Packet Radio Service) module, to connect via GSM to the world wide web and send the gathered data to the ERP's database.

An important feature of the data acquisition system is the use of wireless temperature sensors, specially developed to this project, to allow the easy implementation of the system in any existing distribution fleet. As vehicles here are mostly trucks with triple refrigerated cabins or chambers, each one with its unique temperature characteristics and categories (frozen, chilled and ambient/dry) and certified for food transportation, it is necessary to use wireless sensors to not violate the walls of the chambers by installing a wired communication with the sensors.

Hence, the ZigBee protocol was chosen to allow the wireless interface between the microcontroller unit and the sensors.

The ZigBee devices use a Coordinator/End-Device configuration, that allows to put the ZigBee on the sensor nodes into a sleep mode. When the ZigBee End-Device (sensor) wakes up, it sends the temperature data to the coordinator. This method allows battery saving while the ZigBee is asleep.

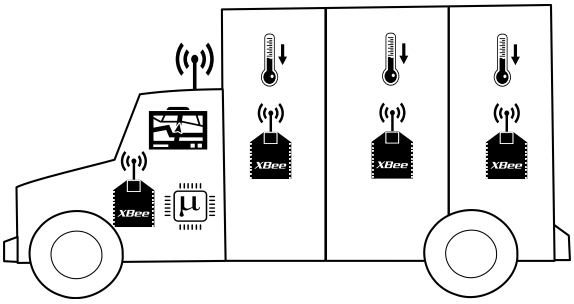

Figure 8: Diagram of the vehicle layout showing the sensors, wireless communication and users interface.

Moreover, each external temperature module has a ZigBee device with four ADC (Analog to Digital Converter) inputs, where two are used to acquire the temperature information from an integrated circuit (IC) temperature sensor, and a signal of the power-supply voltage.

The temperature sensor is the MCP9700 (Microchip, 2014) temperature sensor, which allows acquiring temperatures in the defined range with a low cost.

The voltage signal is acquired to allow monitoring the battery charge available of the power-supply batteries.

As a long battery life is required and the operating environment can have extreme temperatures of -20^{o}C to +60^{o}C, two 3.2 volts LiFePo4 (lithium iron phosphate) type batteries were used, which ensure long lifetimes.

Note that these rechargeable batteries, although they have somewhat lower energy density than other common batteries found in consumer electronics (e.g., LiCoO2 type), offer longer lifetimes, better power density (the rate that energy can be drawn from them) and are inherently safer when working at the considered temperature range.

The data acquired by the data acquisition system, is sent to the ERP's database to be used by the *i3FR-Opt* module. It is also stored in the data acquisition module in a SD (Secure Digital) memory card, to ensure that it is not lost during field operations. Moreover, dedicated printed circuit boards for both, the main control module and the temperature sensor module were specially developed and were already tested in real vehicles, sending the acquired information to the ERP's database, through the Internet.

Figure 8 presents a diagram of the vehicle layout showing the sensors, wireless communication and users interface.

4 CONCLUSIONS AND FUTURE WORK

This work presents the steps used to implement a multi-layered architecture to integrate a SAGE ERP X3 with the developed VRPTW optimization software and the data acquisition module.

Thought to provide scalability, the optimization system was designed to include several services, namely: an optimization module – *i3FR-Opt*, a maps module – *i3FR-Maps*, a database module – *i3FR-DB* and a hub/communication module – *i3FR-Hub*. Among other things, these modules are prepared to: (1) deal with dynamic scenarios, where new demands may be integrated during the optimization process; (2) acquire geographical data from several sources; (3) accept several optimizers services; and (4) maintain the company's core ERP software.

On the other hand, the data acquisition system can be used to obtain important information about the transported goods, namely: transportation temperatures and humidity. The system is modular and easy to install avoiding major physical interventions on the vehicles.

The overall system is in an integration and test phase, where very promising results were achieved.

As future work, among other things, relevant information from the data acquisition module should be further integrated with the routing system (e.g., compare the real time taken in each route to validate the values returned by Google Maps and OSRM). The amount of data retrieved by the acquisition system also indicates the need for a big data analysis module, in order to properly take advantage of the all system.

ACKNOWLEDGEMENTS

This work was partly supported by project *i3FR*: Intelligent Fresh Food Fleet Router – QREN I&DT, n. 34130, POPH, FEDER, the Portuguese Foundation for Science and Technology (FCT), project LARSyS PEstOE/EEI/LA0009/2013. We also thanks to project leader X4DEV, Business Solutions [http://www.x4dev.pt/].

REFERENCES

Abousaeidi, M., Fauzi, R., and Muhamad, R. (2011). Application of geographic information system (gis) in routing for delivery of fresh vegetables. In *Humanities, Science and Engineering (CHUSER), 2011 IEEE Colloquium on*, pages 551–555. IEEE.

Ambrosino, D. and Sciomachen, A. (2007). A food distribution network problem: a case study. *IMA Journal of Management Mathematics*, 18(1):33–53.

Cardoso, P. J. S., Schütz, G., Mazayev, A., and Ey, E. (2015). Solutions in under 10 seconds for vehicle routing problems with time windows using commodity computers. In *Proc. of the 8th International Conference on Evolutionary Multi-Criterion Optimization, EMO'15*, page (Accepted for publication).

Chen, H.-K., Hsueh, C.-F., and Chang, M.-S. (2009). Production scheduling and vehicle routing with time windows for perishable food products. *Computers & Operations Research*, 36(7):2311 – 2319.

Faulin, J. (2003). Applying MIXALG procedure in a routing problem to optimize food product delivery. *Omega*, 31(5):387 – 395.

Glover, F. and Laguna, M. (1999). *Tabu search*. Springer.

Hsu, C.-I., Hung, S.-F., and Li, H.-C. (2007). Vehicle routing problem with time-windows for perishable food delivery. *Journal of Food Engineering*, 80(2):465–475.

JSON (2014). Javascript object notation. http://www.json.org/. Accessed september 28, 2014.

Logvrp.com (2014). Logvrp.com. http://logvrp.com. Retrived on Nov. 27, 2014.

Microchip (2014). The mcp9700 temperature sensor, datasheet. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022289. Accessed December 1, 2014.

MongoDB, I. (2014). Mongodb. http://www.mongodb.com/. Accessed 28/10/2014.

Newronia.com (2014). Newronia.com. http://en.newronia.com. Retrived on Nov. 27, 2014.

Optimoroute.com (2014). Optimoroute.com. http://optimoroute.com. Retrived on Nov. 27, 2014.

optrak.com (2014). Optrak.com. http://optrak.com/. Retrived on Nov. 27, 2014.

OSRM (2014). OSRM – Open Source Routing Machine. http://project-osrm.org/. Accessed september 28, 2014.

Osvald, A. and Stirn, L. Z. (2008). A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of Food Engineering*, 85(2):285 – 295.

Redmond, E. and Wilson, J. R. (2012). *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement*. Pragmatic Bookshelf.

Richardson, L. and Ruby, S. (2008). *RESTful web services*. " O'Reilly Media, Inc.".

Routyn (2014). Routyn. http://www.routyn.com/. Retrived on Nov. 27, 2014.

SAGE, E. X. (2014). Sage erp x3. Accessed september 28, 2014.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Tarantilis, C. and Kiranoudis, C. (2002). Distribution of fresh meat. *Journal of Food Engineering*, 51(1):85 – 91.