Prototype for Importing Routes Generated for Realistic Simulation of Vehicles

John Velandia*, Holman Bolívar**

Abstract — Daimler FleetBoard offers telematic services by means of a special hardware installed in customers' vehicles to collect and send data to the FleetBoard's Service Centre (FBSC) platform. The quality assurance and testing department guarantees that the telematic services meet their purpose and no failures exist in the system. In that way, software that simulates vehicles' behaviour is required for testing the functionalities of FBSC; however, this software uses simulated data instead of real data.In addition,the process of creating routes to simulate a tour is manual. Thus, this paper designs, implements and evaluates a prototype as mechanism of importing routes generated by real vehicles to the simulator's database, so that realism is achieved to tackle the lack of reality. Consequently, an evaluation of the prototypical implementation is considered to guarantee the proper operation along the presentation, business and data layers.

Keywords — Architecture and software engineering, programming, Web Map Services, Quality and testing simulations, and performance analysis.

I. Outline and Background

FleetBoard's products and services are offered to customers through a communication platform supported by telematic hardware installed in vehicles. This platform is composed of three fundamental parts for collecting, processing and sending data, where the first part constitutes all vehicles from customers, the second one is the FleetBoard Service Centre (FBSC) and the last one is the customer software interface. Additionally, these three points contain a well-defined workflow that allows them to interchange data one to another Figure 1 illustrates the communication platform.

Every vehicle has installed the TiiRec which is a hardware configured in the vehicle to establish the communication and send data to FleetBoard Service Centre. The TiiRec contains a GSM/GPRS modem for communication over mobile networks, and the Global Positioning System (GPS) receiver for the vehicle position tracking.

The FleetBoard Service Centre is responsible for receiving data from vehicles and providing data to customers. Communication is performed using a private protocol. The data are stored in a distributed database which increases its availability and enhances the performance of responses to customers.

Customers access their services throughout internet using a Web GUI or Web services interfaces, e.g., customers see in real time where their vehicles are.

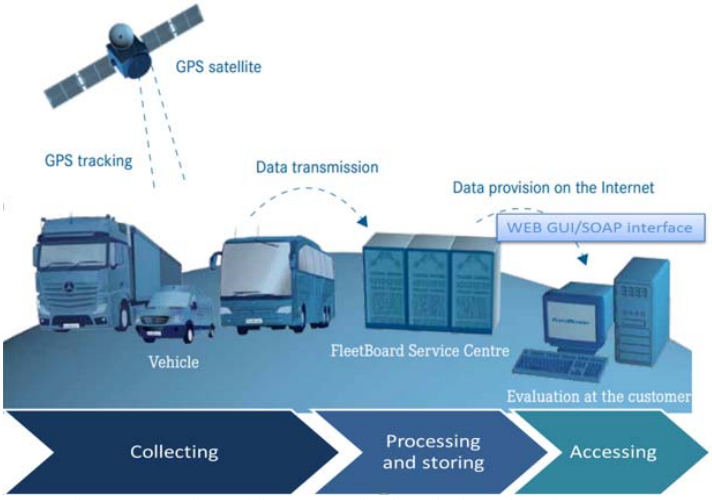

Figure 1 Overview of the communication platform, based on [1]

A. The architecture

The architecture comprises FBSC to receive messages from real vehicles and provide services to customers, and Telematic Platform (TP) to collect data in vehicles and send it to FBSC. In addition, a parallel architecture exists with the purpose of testing and simulating current and new features in FBSC. The benefits of having this parallel system encompasses reduction of unexpected behaviours or errors in the production environment, more quality in the services provided to customers, and real trucks' hardware are not used for testing purposes, that also minimize costs.

The parallel architecture contains Integration Test System (ITS) that substitutes FBSC, and LiveVehicleSim that replaces real vehicles. ITS is used for testing FBSC and LVS to simulate vehicles. Running both systems allows measuring of FBSC's performance and guarantees that FBSC's functionalities work properly with high quality. Figure 2 depicts the architectural overview.

The process of adding new features on FBSC considers first a deployment of these features in ITS, then old and new features are tested using several testing tools, from which LVS is also included. Thus, if results are successful the new function-

* Facultad de Ingeniería, Universidad Católica de Colombia, Bogotá, Colombia, javelandia@ucatolica.edu.co
** Facultad de Ingeniería, Universidad Católica de Colombia, Bogotá, Colombia, hdbolivar@ucatolica.edu.co

978-1-4799-6717-9/14/$31.00 ©2014 IEEE

alities are deployed in real environment and customers are willing to use them. In case of having errors or weird behaviours new functionalities are corrected and a new test process starts.

In this paper, the development and implementation of the prototype consider the testing and simulation environment. In that way, LVS is used to reach more realism in the simulations, and ITS will be utilised to receive messages and to import historical data from real vehicles to LVS.

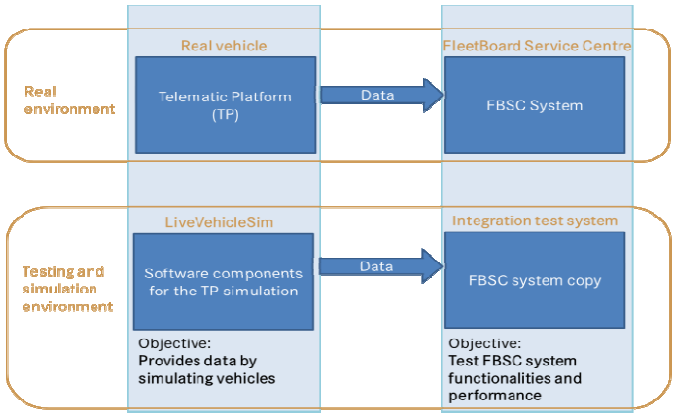

Figure 2 General system architecture overview

II. PROBLEM STATEMENT AND OBJECTIVES

A. *Problem statement*

The department for quality assurance needs to test the functionality of the entire platform based on software using existing or generated data. In the past, data for testing were generated or real customers data were copied from the production environment before the tests started. One problem with the existing approach is that the generated data are not close to customer behaviour and the second problem is that the real customers data are static, thus tests do not represent a running system with up to date data coming in continuously.

The first approach to solve these problems is the LVS, which generates predefined tours for vehicles and so provides the live data needed for more realistic tests. The actual process of simulating a journey, also known as tour simulation, comprises a vehicle with its driver, thus, a virtual vehicle drives from one place to another, following a route and sending messages to central servers. In this way, data are processed and stored as it came from real vehicles, e.g., data such as speed of vehicle, fuel consumption, etc.

A truck driving all time just for testing purposes is not beneficial for the company; this represents additional costs and time. Thus, tours simulations are performed daily to test functionalities of the FleetBoard central servers. These simulations also help to minimize the impact of unexpected errors and ensure the correct running of the new and old functionalities offered to customers.

However, current simulations use routes that are not close to customer behaviour, because they are created using Google Maps. Furthermore, the creation of routes is a manual process. In detail, creation of routes comprises the setting of a route on Google Maps. The result from Google is transformed into GPS Exchange Format (GPX) to acquire a representable route with coordinates in between. Finally, coordinates are copied in the Web Graphic User Interface (GUI) and stored in the database of LVS.

As second approach, this paper should provide new functionalities over LVS that incorporate real routes in tour simulations, and an automatic process to create routes using Web Maps, e.g., Nokia Maps, Google Maps, etc. In this way, simulations will be close to reality, and manual processes are eliminated.

B. *Objectives*

- Develop a refined concept and design based on the analysis of the data source for the simulation and the simulation itself in detail.
- Implement a prototype for the concept. The prototype has to enable the simulation of all fleets at the same time using the previously developed concept and algorithm.

The Structure of this paper is as follows: Section II presents the problem statement of the software simulator. In Section III an analysis, design and solution is proposed for addressing the goals of this paper. Section IV presents the system's implementation. Section V contains test and validation of the prototype. In Section VI it is presented the proposal results and the discussion. Finally Section VII includes the conclusions and future work.

III. ANALYSIS, DESIGN AND SOLUTION

A. *Analysis for incorporating real routes in tour simulations*

The process of running a tour simulation consists of generating data close to the reality for acquiring behaviour similar to real vehicles, with the objective of testing functionalities of FBSC. Currently during the execution of a tour simulation LVS follows routes manually created using Google Maps to simulate vehicles' movement. These routes do not represent any relation to the reality, i.e., none vehicles have driven over these routes, so that, actual simulations lack of realism, and the cause lies on using GoogleMaps as source to create and import routes.

In reality, every vehicle follows unique routes, even if a vehicle follows the same roads daily with same initial and end point, they differ in details, because a waypoint is not always collected at the same position and at the same time by the TiiRec, it varies from one to another. In that way, there must be more routes than vehicles in a simulation, which is not true for LVS. For this reason, it is essential to find a solution that provides the amount of data necessary to generate tours automatically with unique routes.

The following approaches encompass the problem statement defined in section II, from which it is stated the lack of realism of tours simulations, and the non-existence of an automatic process to create amounts of routes that satisfies the number of available vehicles.

B. Design and solution

The solution consists of the optimization of importing routes based on drawing a line as input of information using Web maps as first activity to define a route, this activity is performed in LVS. The final operation consists of sending a request to save the coordinates that were set in the previous activity, and then, an output is displayed as confirmation message on Web GUI to corroborate whether the operation was successful or not.

The advantage of this solution is that manual activities are removed and a considerable minimization of time is achieved when the execution of importing routes is performed; since only one system operates all the activities without the intervention of manual tasks. However, a further analysis and evaluation is needed to define the provider of the map API before this is integrated into LVS.

C. Evaluation of different WMS providers

According to Schmidt and Weiser in Online Maps with APIs and Web Services [2] and Błażej [3], companies such Google, Microsoft, OpenStreetMap and Nokia are the most relevant in the market of WMS. This is due to the success of their services and the capacity to handle their services and support. Thus, these four main providers are part of the evaluation for integrating a map API with its WMS into LVS.

		Company's Map service			
Criteria	**Weight**	**Google**	**Bing**	**OpenStreet Map**	**Nokia**
Zero cost of investment	10				✓
Functionality	9	✓	✓	✓	✓
Reliability	8	✓	✓		✓
Usability	7	✓	✓	✓	✓
	Total	24	24	16	34
Selection					✓

Table 1 Evaluation and selection of the WMS Provider

Due to the results of Table 1, Nokia is the most suitable provider for a WMS and map API. This decision is based on the results from Table 1, from which Nokia reached 34 points out of 34 along different providers, despite all of them are supported by the same principle, which consists of using a central server called Map Tiles Server [4], a distributed database system and Web browser to make request using a map API.

IV. IMPLEMENTATION

The new functionalities are developed in Java using Integrated Development Environment (IDE) Eclipse [5]. For the Application Server and Servlet Container is Apache Tomcat [6]. The RDBMS is MySQL [7]. Maven is used as tool for Project Build Manager [8] and SVN is used for version control on the file level [9].

Considering the MVC pattern [1] and its advantages [2] [12], the view is built it up using Java Server Faces(JSF), including a HTML tag library, for the user interface (UI) components, JSF core tag library to customize actions and RichFaces tag library for easily integrating Asynchronous JavaScript and XML (AJAX) features into the application. The model layer uses Apache Open JPA implementation for the Java Persistence API specification [10].

This section describes the new behaviour of CRUD operations in the routes management Web GUI using routes from FBSC.

1) Creation of routes

The objective of this operation is to create real routes using the Web GUI. For every execution of this operation a set of routes is created in LVS, the size of the set depends on the number of vehicles that belongs to a fleet's selection in the Web GUI. For every execution users only see a confirmation message whether operation was successful or not, which is similar to the previous mechanism to create routes. However, they differ from input data and the algorithms.

2) Update of routes

Update operation consists of bringing up to date data regarding real routes. Thus, considering a selection of a parent route and any modification of its name and fleet, this operation removes the set of routes that belongs to the parent; afterwards, the same process of creating routes is applied. This operation is based on the definition of the use case Update routes.

3) Deletion of routes

The aim of this operation is to remove a set of routes that belongs to a parent route selection on the Web GUI. After executing this operation a message is displayed on the Web GUI confirming whether the operation was successful or not. If operation is successful, the selected parent route from the left panel, is also removed, otherwise parent route remains in the Web GUI. This functionality is based on the use case delete.

4) Retrieving routes

This operation retrieves routes from LVS' database every time the Web GUI is requested. The previous mechanism retrieved routes one by one. The new implementation keeps the previous mechanism, because this is used for routes that come from WMS (Web Map Service) provider (Google Maps). In addition, to retrieve routes that come from FBSC the new implementation displays only parent routes which represent sets of routes. Thus, the Web GUI displays all routes that are created using WMS provider, and parent routes. This operation is supported by the use case Display.

5) Nokia Maps

This section describes the new functionality that allows creating routes automatically from WMS provider. This functionality runs on the routes administration Web GUI. Create and retrieve operations are described, while delete and update operations are not treated, because they are not modified.

Technically this Web GUI integrates Nokia map API to interact with the Web GUI and WMS [2]. Consequently, API's functions and a tailored algorithm are used to import and load waypoints automatically. This functionality is based on the use case configure Nokia API and create routes using Nokia API [14].

Using Nokia JavaScript API users interact with a Web map in the Web GUI by setting the initial and final points of a route, thereafter an asynchronous request is sent it to Nokia's servers, and the response is formatted to a route which is display in the Web GUI (see Figure 3).

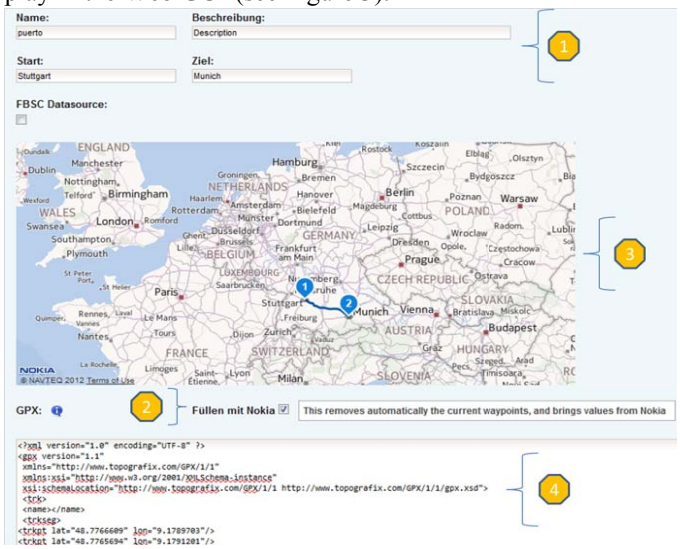

Figure 3 Interaction of Nokia WMSand the routes administration Web GUI

V. TEST AND VALIDATION

In this section the developed prototype is evaluated using testing methodologies and tools. Tests' results are validated based on: (1) functional requirements and non-functional requirements according to the prototype's performance.

A. *Funcional requirements*

In this section JMeter is configured with scripts to perform CRUD operations in the administration of routes Web GUI while every operation is executed once and its response is evaluated.

Tests regarding the importation of routes and CRUD operations that involve routes are running properly thus, they meet the functional requirements. Table 2 corroborates the success of tests by observing thread groups 1-2 retrieving, 2-2 creation, 3-2 update and 4-1delete.

Tests focus on sending requests to Nokia Maps server by means of the Nokia maps API, and the responses that contain huge sets of waypoints that define routes. Functional requirements are met according to the success of thread group 1-1, 2-1, and 3-1 from Table 2.

Thread Name	Operation	Status
Group 1-1	Retrieve routes using Nokia WMS	Success
Group 1-1	Retrieve routes using Nokia WMS	Success
Group 1-2	Retrieve routes using FBSC' database	Success
Group 2-1	Creation of routes using Nokia's waypoints	Success
Group 2-2	Creation of routes using FBSC's waypoints	Success
Group 3-1	Update routes using Nokia WMS	Success
Group 3-2	Update routes using FBSC's waypoints	Success
Group 4-1	Delete operations using FBSC's waypoints	Success

Table 2 Results of the CRUD operations, using HTTP request

B. *Non-functional requirements*

This section provides the evaluation of non-functional requirements to guarantee an adequate performance of the prototype in terms of responsiveness and stability. Response time analysis is applied to validate the responsiveness [12]. The throughput to validate the stability [12]. The following methodology is defined to evaluate the performance using JMeter.

1) *Methodology*

JMeter is used in order to evaluate the performance of the implementation. The performance metrics that are considered comprise throughput to measure requests per second to the server [12], and the average response time, which is the mean value of the elapsed time between a request to the server and the receipt of the response in the browser [12]. These metrics are useful, because the acceptability of a test is obtained from them. Consequently, if test is accepted, responsiveness and stability of the system based on the new functionalities are guaranteed.

The performance of an application comprises two criterion for acceptance, the response time, which is user concern, and the throughput as business concern [13]. In that way, the response time is evaluated using the confidence interval analysis with thresholds determined by human behaviour, and the throughput is directly compared to business constraints

Because average response time is sometimes a misleading measure [11], an additional evaluation is performed using confidence interval analysis method to obtain a measure close to reality. This case is presented when response time values are far from the average response [12], [16]. An example is provided in Table 3, from which Test 1and Test 2 show average response time and standard deviation that result after running every test. Clearly, Test 1 provides a misleading average, because every sample data are far from the average, e.g., x1=16s is far from 5s, which is called variability. On the other hand, Test 2 provides an ideal scenario, since sample data is close to the mean value, e.g., x4= 5s is close to 5s.

	Test 1 (Seconds)	Test2 (Seconds)
	x1= 16s	x1= 5s
	x2= 2s	x2= 4s
	x3= 1s	x3= 6s
	x4= 1s	x4= 5s
Average resp. time	5s	5s
Standard deviation	6,36	0,70

Table 3 Real and misleading average response times

Confidence interval analysis solves the problem of misleading average response time. This analysis estimates the variability of the sample data, and provides an average response time close to reality based on a predefined probability or interval. In addition, the standard deviation is used for the calculation of the confidence interval, and also for determining whether the average response is misleading, e.g., if Standard deviation is low than the average response time, then average is accurate, otherwise misleading [14]. This paper will adapt the response time analysis for all evaluations to obtain more accurate results, even if the standard deviation is lower than the average response.

The response time analysis is divided in three sections: a) Definition of the confidence interval that provides details about the mode of estimating the confidence interval and conditions. b) The well-known thresholds for response times that Web applications should consider. c) Validation of results, from which tests results are validated by comparing them to the thresholds. These three sections are briefly described in the following paragraphs:

a) Definition of the confidence interval

The acceptability of the response time metric is determined using the confidence interval analysis method, which is based on the Central Limit Theorem, that states: if a representative part from a group has an average distribution (μ) and standard deviation (σ), and then, for at least 30 samples, the sampling distribution has an approximate normal distribution [14]. In this case, JMeter provides the average response time that represents μ, standard deviation that represents σ, and the number of samples considered is 31. For further details about the central limit theorem refer to the literature [14].

$$\text{Confidence interval} = X \pm Z_{\frac{\alpha}{2}}\left(\frac{\sigma}{\sqrt{n}}\right) \quad \textbf{(6.1)}$$

Formula 6.1 is used to analyse the confidence interval. From which, X defines the average of the sample (μ), $Z_{\frac{\alpha}{2}}$is a fixed value that comes from Table 4 that represents the probability of samples included into the calculated confidence interval (average response time). In this paper the defined probability is 95%. σ defines the standard deviation, n is the size of the sample, and the confidence interval represents the level of certainty of the sample.

Confidence interval level	Z
0.90	1.645
0.92	1.75
0.98	2.33
0.99	2.58

Table 4 Z confidence level intervals

b) Thresholds for response time

These are used for determining if the response time of operations is optimal. These thresholds are derived from research regarding human perceptual abilities and brain behaviour [15].This paper applies these thresholds to give a criterion after calculating the confidence interval.

Threshold	Response time	Description
1	<0.1 s	Users do not notice a delay.
2	0.1 - 1 s	Users will notice the delay but this won't interrupt their work flows.
3	1 s - 10 s	Usersactively wait for a response and consciously consider this an interruption.
4	>10 s	Users lose focus and start doing something else.

Table 5 The well-known thresholds for response times using Web applications

c) Validation of results

Table 6 contains results generated by JMeter after running tests for CRUD operations, and also results after applying confidence intervals. The values for retrieving routes operation are replaced into the Formula 6.1, from which the confidence obtained is equal to 0.37s (see Formula 6.2). Thus, according to the threshold 2 from Table 5, the response time is acceptable for the retrieve operation. Similarly, the confidence interval for the creation is 1.08 s, update is 1.76 s, and delete equals 0.13s, which means, that these results are acceptable considering threshold 3.

Operation	Average response time (sec)	Throughput (req/time unit)	Confidence interval (sec)	Standard deviation (sec)
Retrieve	0.347	4.9 req/sec	0.37	0.12798
Create	0.736	10,6 req/min	1.08	1.00397
Update	1.462	11,3 req/min	1.76	0.86025
Delete	0.604	6.9 req/min	0.13	0.38563

Table 6 JMeter Summary report after running tests for CRUD operations over routes

$$Confidence\ interval = 347 \pm 1.96\left(\frac{127.98}{\sqrt{62}}\right) = 378{,}85\text{ms} = 0.37\text{ sec} \quad \textbf{(6.2)}$$

2) Throughput analysis

Throughput metric is obtained dividing number of request by total of time (end time of the last sample – first sample time) [16]. After running the test, the throughput for the *retrieve operation* is 4.9 requests per second (see Table 6).This is an acceptable result considering that in the worst case the maximum number of users performing this operation simultaneously does not exceed this result. These CRUD operations associated to routes are part of the configuration of LVS, so that, users do not use this functionality frequently. Similarly, the result for *creation* is 10.6 requests per minute, *update* is 11.3 requests per minute and *delete* equals 6.9 requests per minute.

In addition, throughput metric is also used for measuring where WAS (Web Application Server) reaches its overload point, however based on Table 6, the behaviour of the WAS does not reach saturation in neither of CRUD operations.

VI. RESULTS AND DISCUSSION

Considering objectives, section II, a set of pre-configured tests run on CRUD operations to validate and evaluate the correct running of the prototype. Tests were based on sending data by means of the Web GUI, and receiving it using JMeter. Results regarding functional and non-functional requirements are founded on JMeter's reports.

Results of functional requirements are analysed and evaluated from two perspectives. The first perspective encompasses the collecting of headers responses which are received by JMeter after server processes requests. Success of an operation is determined by these headers responses, e.g., headers' content with value equals 200, means that the operation requested was executed successfully.

The second perspective is focus on comparing expected

data to real data which are generated after performing any CRUD operation. Thus, if expected data equals real data, then the requested operation is successful, otherwise unsuccessful.

The performance of the prototype is based on two criteria: (1) response time, which is the elapsed time since a request is sent to the server, and subsequently a response is received by the user. (2) Throughput, which represents the number of requests accepted by servers during a time unit, thus this criterion determines if servers reach an overload point, i.e., servers do not process requests after a certain number of requests.

In this paper, the response time of CRUD operation is acceptable if it is less than 10 s, which is supported on the well-known thresholds for response time, Table 6. In addition, the throughput is acceptable along as this does not reach an overload point.

The result of the prototype validation in this section demonstrates that all objectives defined for this paper are successfully achieved. Functional tests support the proper running of the prototype considering the objectives. Reliable results regarding performance analysis ensure the correct behaviour of the application on certain conditions. These results guarantee a simulation close to the reality based on mass data generation

VII. CONCLUSIONS AND FUTURE WORK

This paper provides a background concerning the components needed to perform tour simulations, including the internal protocol to establish communication between FBSC and LVS.

Two concepts are defined to tackle the two problems defined in this paper. The first concept analyses the lack of realism that is caused by using routes from Google Maps. The solution defined in this concept consists of replacing Google Maps service by FBSC's database to obtain simulation with more realism, since these routes are generated by real vehicles. In addition, this concept provides an algorithm that performs the creation of routes automatically based on functional requirements previously defined.

The second concept emphasizes in the optimization of the current process of importing routes using WMS. The problem defined in this section comprises the manual tasks to define a route with valid waypoints, and the solution consists of integrating Nokia Maps into LVS. Several approaches were defined to provide the best optimization, and an analysis and evaluation regarding the WMS providers were considered to come up with the most suitable provider, Nokia.

The implementation of the prototype is based on the concepts and the functional requirements. JMeter tool is used to evaluate the prototype with the functional requirements. The results of this evaluation show that the prototype runs as it was specified. Non-functional requirements are evaluated using performance analysis considering response time and throughput metrics. The results after evaluate these non-functional requirements are successful.

Further future works involve the verification and evaluation of the data completeness, because routes imported from production environment sometimes are empty. These sort of data are come up from human errors, e.g, drivers sometimes forget to put the identification card in the TP, this means that a range of waypoints cannot be imported due to missing initial or final points of real routes. Another issue is with regard the mode of storing waypoints, establishing a manner to differenciate which vehicles store waypoints in tables gpsdata and gpstracedata, may improve the response time and throughput of the prototype.

REFERENCES

[1] A. Leff and J. Rayfield, "Web-application development using the Model/View/Controller design pattern Fifth IEEE International," in Enterprise Distributed Object Computing Conference, 2001, p. 127.

[2] S. Li and L. Sun, "Advantages analysis of JSF technology based on J2EE," in Computer Science and Service System (CSSS), 2011.

[3] Daimler, FleetBoard. Daimler FleetBoard GmbH. [Online]. http://www.fleetboard.com/info/en/company-portrait.html

[4] Schmidt M and Weiser M, Online Maps with APIs and WebServices. Berlin, Germany: Springer, 2012.

[5] B Ciepłuch, R Jacob, P Mooney, and A Winstanley, "Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps," Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences, vol. 20-23rd, p. 337, September 2010.

[6] A.J Turner, Introduction to Neogeography.: O'Reilly Media, Inc, 2006.

[7] Eclipse. (2014, January) Eclipse, Eclipse. [Online]. http://www.eclipse.org/

[8] Apache. (2014, June) Tomcat Apache. [Online]. http://tomcat.apache.org

[9] Oracle. (2014, June) MySQL. [Online]. http://dev.mysql.com/downloads/mysql

[10] Apache. (2014, June) Apache Maven Project. [Online]. http://maven.apache.org/

[11] Apache. (2014, June) Subversion, Apache Subversion Project. [Online]. http://tortoisesvn.net

[12] Y. Wang and Q. Wu, "Performance Testing and Optimization of J2EE-Based Web Applications," in Education Technology and Computer Science (ETCS), 2010, p. 683.

[13] Apache. (2014, June) Apache OpenJPA project. [Online]. http://openjpa.apache.org

[14] J. Fu, Y. Li, and K. Zhu, "Research the performance testing and performance improvement strategy in Web application," in Education Technology and Computer (ICETC), 2010, p. 328.

[15] P. Bansode, S. Barber, C. Farre, and J. Meier, Performance Testing Guidance for Web Applications.: O'Reilly Media, Inc, 2007.

[16] M. Haklay and C. Parker, Web mapping 2.0: the neogeography of the GeoWeb, 2011th ed.: Geography Compass, 2008.

[17] O. Johnson, Information theory and the Central Limit Theorem. London, England: Imperial College Press, 2004.

[18] J. Nielsen, Usability Engineering. London, England: Academic Press, 1993.

[19] Apache. (2014, June) Apache JMeter. [Online]. http://jmeter.apache.org/

[20] G. Di Lucca and A. Fasolino, "Testing Web-based applications: the state of the art and future trends," in Computer Software and Applications Conference. COMPSAC, 2005, pp. pp.65,69 Vol. 1.