DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Choreographing Traffic Services for Driving Assistance

EFTHYMIOS NEROUTSOS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

KTH Royal Institute of Technology
School of Information and Communication Technology
MSc in Communication Systems

Choreographing Traffic Services for Driving Assistance

A thesis project within

Efthymios Neroutsos, eftner@kth.se

Industry Supervisor: Lei Chen, Viktoria Swedish ICT

KTH Examiner: Fredrik Kilander
KTH Supervisor: Anders Västberg

Abstract

This thesis project presents the web service choreography approach used for the composition of web services. It leverages the CHOReVOLUTION platform, a future-oriented and scalable platform, that is used to design and deploy web service choreographies. By using this platform, a use case that falls into the ITS domain is developed. This use case highlights the benefits of the web service choreography when used for the development of ITS applications. The necessary web services are designed and their interactions are defined through a choreography diagram that graphically represents how the services should collaborate together to fulfill a specific goal. By using the choreography diagram as input to the platform and by registering the web services on a web server, the choreography is deployed over the platform. The resulted choreography is tested in terms of services coordination. It is demonstrated that the platform can generate specific components that are interposed between the services and are able to take care of the services coordination for the use case created. Moreover, the execution time required to complete the choreography is measured, analyzed and reported under different conditions. Finally, it is shown that the execution time varies depending on the data that the services have to process and that the processing of huge data sets may lead to high execution times.

Keywords*: Web Service Choreography; Choreography Diagram; Intelligent Transportation Systems*; *Driving Assistance*

Sammanfattning

Detta examensarbete behandlar hur man med hjälp koreografering av webbtjänster kan komponera webbtjänster. Det använder sig av CHOReVOLUTION plattformen, en framåtblickande och skalbar plattform, som används för att designa och verkställa koreografering av webbtjänster. Med denna plattform skapas ett användningsfall inom ITS-området. Detta fall belyser fördelarna med webbtjänskoreografi i samband med utveckling av ITS-applikationer. De nödvändiga webbtjänsterna designas och deras samspel definieras genom ett diagram för koreografin, som på ett grafiskt vis presenterar hur tjänsterna skall kollaborera för att nå ett specifikt mål. Genom att mata plattformen med data från diagrammet, och genom att registrera webbtjänster på en webbserver, verkställs koreografin. Med resultatet testas koordineringen av tjänsterna. I detta examensarbete visas det att plattformen kan skapa specifika komponenter som interagerar med tjänsterna, samt sköta koordineringen av tjänster som krävs för detta användningsfall. Exekveringstiden mäts, analyseras och rapporteras under flera olika omständigheter. Det demonstreras också att exekveringstiden varierar beroende på den data som tjänsterna måste behandla, och hur behandlingen av mycket stora datamängder kan leda till långa exekveringstider.

***Nyckelord**: Webbtjänstkoreografering, Koreograferingsdiagram, Intelligent Transportation Systems, Körhjälp*

Acknowledgements

I would first like to thank my thesis supervisor, Lei Chen, who throughout the process of conducting this thesis was always there for me, willing to give me directions and answer my questions. His positive and optimistic attitude was a great motivation for me. I feel really grateful for the time he dedicated to this thesis project.

I would also like to thank Cristofer Englund for giving me the chance to conduct my thesis at Viktoria and be, even for a short time, part of the CHOReVOLUTION project.

Finally, I want to express my gratitude to my family and my close friends for providing me with support and continuous encouragement throughout my years of study. ☺

Table of Contents

Abbreviations and Acronyms

ADAS	Advanced Driver Assistance Systems
API	Application Program Interface
BC	Binding Component
BPEL	Business Process Execution Language
BPMN2	Business Process Model and Notation version 2
CD	Coordination Delegate
C-ITS	Cooperative Intelligent Transportation Systems
CoAP	Constrained Application Protocol
CVR	Connected Vehicle Research
FI	Future Internet
GeoJSON	Geographic JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
IDRE	Integrated Development and Runtime Environment
ICMP	Internet Control Message Protocol
IoT	Internet of Things
ITS	Intelligent Transportation Systems
IVIS	In-Vehicle Information Systems
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
ODE	Orchestration Director Engine
REST	Representational State Transfer
SF	Security Filter
SOAP	Simple Object Access Protocol
UTC	Urban Traffic Coordination
WSC	Web Service Composition
WS-CDL	Web Services Choreography Description Language
WSCI	Web Service Choreography Interface
XML	Extensible Markup Language

List of Figures

1. Introduction

During the last decades, it has been a global phenomenon that more and more people are moving to larger cities all over the world. This fact is a further obstacle for the challenges and problems that modern societies and large cities must face and address in order to operate normally and guarantee the safety and well-being of their residents. A basic part of these challenges is the congestion of the transportation network which decreases not only the quality of life but has also a big impact on the environment and the sustainability of the city's ecosystem [1]. While many solutions have been proposed to solve this problem [2], one of the easiest and simplest solutions will always be to allocate drivers in such a way that their combined driving behavior will work together to increase the efficiency within the transportation network.

However, in order to have efficient and reliable control of the transport, efficient systems need to be developed. Systems that can provide a highly accurate estimation of the current traffic flow as well as highly accurate forecasts. Even if accuracy may be less than exact, it is safe to say that it is preferable than no control at all. Cooperative Intelligent Transport Systems (C-ITS) define a system of advanced applications that their aim is to allow vehicles to connect together and exchange messages and information related to traffic management and enhance the awareness of drivers [3]. Thus, the drivers and the vehicles are able to be better informed and use the transport networks in a safer, more comfortable, more coordinated and smarter way.

Urban traffic coordination (UTC) helps to improve the traffic efficiency and to reduce the environmental impact. To develop such an application, real-time data services are required and efficient data and service synthesis methods are needed. This need brings the research within CHOReVOLUTION [4] to practice through integrating smart and efficient Intelligent Transportation Systems (ITS) methods into the future smart city system. UTC applications require a distributed, efficient platform to adapt to the high density and heterogeneous data of the urban transportation system. The platform must be highly scalable and should have the ability to integrate into the Future Internet. With the purpose for large-scale application development, CHOReVOLUTION aims to provide a platform for developing dynamic and secure choreographies, which have the ability to integrate the ITS system into the future smart city system.

In essence, this platform leverages the advantages of the choreography web services architecture, where in the absence of a central coordinator, the

services are able to interact together as peers to fulfill a specific goal. Thus, we are moving from a centralized environment to a more distributed one, where the business logic and the related coordination is not concentrated into a central controller but is distributed among the participating services.

1.1 Problem

Traffic applications nowadays require real-time data services from multiple providers and domains. Several data sources that already exist, like Google Maps [5] or the Swedish Road Administration [6], can be leveraged to define new services that can be combined in order to produce a complete and accurate traffic application. Choreography provides a method to collaborate those services through defining interactions between them so that once the choreography is synthesized, enacted and the application is executed, each service knows when and to whom it should interact without global coordination.

However, automatic synthesis of choreographies to realize a traffic application is challenging. In the absence of central coordination, it is highly important to define correct and clear interactions between the services logically with consideration on the heterogeneity of service. This should be done in order to avoid undesirable behavior of the services, which would result in having no control of the communication between the choreography participants and thus devastating results on the end user application.

As a result, the main problem for a choreography-based traffic application is to develop an automation platform that is able to specify the interactions of heterogeneous services with choreographies, synthesize the choreographies securely and automatically, deploy the related services and choreographies over a cloud system, thus enabling the final application.

1.2 Purpose

The purpose of this thesis is to leverage a choreography-based traffic application and the related web services and deploy them over the CHOReVOLUTION platform, a platform that aims to automatically synthesize dynamic and secure choreographies. This platform targets future Internet-of-Things with large scale and heterogeneous web services, and seems to be highly suitable for connecting different elements of the transport system like road users

such as cars, infrastructures such as traffic lights, road sensors, as well as static and dynamic traffic information such as road conditions, road works, bridges opening statuses and congestion statuses. Through demonstrating applications within the ITS domain based on the choreography platform, it is possible on the one hand to work towards delivering an advanced driving assistance application for efficient and eco-friendly driving, and on the other hand to showcase the feasibility of the platform. The thesis work aims to provide insights on the usage of the platform within the ITS domain that will help to improve the platform thus enabling more complicated functions for urban traffic coordination.

1.3 Scientific Question

By combining the problem statement as well as the purpose of the thesis, specific questions were formed that had to be answered during this thesis. These questions are listed below:

1) Is this specific platform able to automatically produce the required components for the designed use case and choreography?

2) Are the generated components able to accurately coordinate the interactions between the services for the specific use case according to the specific coordination logic; or interactions, which are not desirable, occur leading to unpredictable behavior of the use case? This question should be investigated and answered assuming that:

    i) Any external interference is absent and does not occur during the choreography execution.
    ii) The deployment environment and the services are stable and cannot be altered while the choreography is running

3) What are the different factors that may affect the latency of the designed choreography?

4) Are there any of these factors that can be controlled by the developer of the use case in order to decrease the latency inserted by them and improve the overall latency of the system and the choreography?

5) What is the overall latency (or execution time) of the system for the use case that will be demonstrated and how much does it vary for different choreography executions?

6) Which are the main contributing factors in the variability of the choreography's execution time?

1.4 Goal and tasks

The goal of the thesis was to design and build a use case in order to showcase from the transport domain the benefits of a service choreography approach towards the development and management of complex interactions in the future Internet of Services. It was also used as a use case to showcase the platform and demonstrate that it is capable to solve the complex problem of the services coordination mentioned above with correct coordinations between different services for the deployed use case. Moreover, by performing the required measurements, the overall execution time of the choreography was measured in order to find out which are the factors that affect the latency and how each of them contributes to it.

This general goal can be subdivided into more detailed sub-goals and fulfilled with concrete tasks as follows.

- To investigate possible external data sources that may be used in order to provide routing and real time traffic information data, the type of the data that is provided and how to access this data,
- To get familiar with the choreography concept and design the choreography and the choreography diagram to be used by the platform,
- To implement necessary services that will be used to gather and process data from external services,
- To deploy the web services choreography over the platform for automatic synthesis, enactment and execution, thus realizing the traffic application.

What is more, the whole development process and the final results of the deployment were used to show the validity of the platform for the proposed use case and check initially the performance of the choreography-based methods for traffic application development. This resulted into discussions regarding the platform and the benefits of choreography-based methods and its applicability within the ITS domain.

1.5 Sustainability

The environmental impact of the work described on this thesis can be described as positive. In general, one of the reasons the ITS domain is beneficial, is that it is trying to improve the management of the road infrastructure and provide solutions that will reduce traffic jams and will minimize the driving time spent, thus minimizing CO_2 and other harmful emissions and having a positive impact on the environment and the city ecosystem. The work presented in this thesis is moving towards this destination as it aims to provide accurate routing assistance and enable drivers to spend less time on the streets by providing efficient eco-friendly routes taking into account the current traffic situation.

What is more, the companies can be economically benefited from the use of the platform. They will not have to design their own complete web service coordination systems and deploy them over a private cloud infrastructure, but they will be able to simply design the desired web services, specify how these services will collaborate together and deploy them over the proposed platform and over the cloud system that this platform will already be running. It should also be mentioned that these web services can be reusable without having to alter them. This will result in less expenses and thus more profits.

One more aspect that has to be examined when it comes to sustainability is the social impact of this work. One of the aims of the use case that will be developed is to provide the drivers with optimal eco-friendly routes and real time traffic data. This gives the drivers choices of eco-friendly routes, i.e., the routes with the minimal CO2 emission, thus minimizing the environmental impact from driving. In the meantime, with real time traffic data, the drivers will have a better overview of their route and they will be able to avoid areas with huge congestion or roads affected by traffic accidents, making their driving experience more pleasant. Moreover, less driving time means more fuel savings.

1.6 Ethics

Apart from the aforementioned sustainability aspects, there are some ethical aspects that need to be mentioned. First of all, for the purpose of the use case some external data sources need to be used. These sources or Application Program Interfaces (APIs) are developed by third party providers or companies and they are accompanied by terms of use. During the exploitation of them, it needs to be ensured that these terms of use are not violated.

What is more, the users of the application will have to submit personal data like their origin and the destination they want to reach, location data for real time navigation, etc. This data will be collected based on the user agreement and should be kept and processed by the services in order to only calculate and deliver the final result to the end user. Under no circumstances should this data be kept for further use by the services provider, especially without the consent of the end user.

It has to be mentioned that these issues are briefly presented here and will not be investigated further during the thesis. The terms of use for the APIs that will be used can be found on the website of each API provider, like in [7] and [8]. In addition to that, the user agreement for the platform will be defined in detail later, when the exact business model of the platform will be defined, and not during this thesis.

1.7 Structure of thesis

The rest of the thesis is organized as follows.

Chapter 2 describes in detail the research methodology that is followed during every step of the work conducted during the thesis work, from the literature study, problem formulation and solution, to the implementation and the final evaluation.

Chapter 3 introduces the reader to the Intelligent Transportation Systems domain and its different subcategories. Moreover, it presents the two main web service composition methods, web service orchestration and web service choreography, and their differences as well as gives an overview of the CHOReVOLUTION platform, the Business Process Model and Notation version 2 (BPMN2) choreography diagrams and how the choreographies are synthesized over the platform.

Chapter 4 presents the use case that is implemented and deployed over the platform and explains thoroughly every design and implementation choice that is made.

Chapter 5 is divided into two parts. The first part evaluates the services coordination that is achieved from the platform for the use case implemented in the previous chapter. The second part evaluates the use case in terms of the

execution time required to complete the choreography. A number of different series of tests are done in order to draw interesting conclusions.

Finally, chapter 6 concludes the thesis, explains the main limitations faced and suggests future work.

2. Research Methodology

As mentioned previously, the goal of the thesis was to explore the CHOReVOLUTION platform, create a simple use case which comes from the ITS domain, validate the service interactions specified in the choreography and explore the potential benefits of a choreography approach for ITS applications. In order to accomplish this and answer the questions presented in the previous chapter, the research work within the thesis followed a concrete methodology plan as follows.

The research methods that were used were both quantitative and qualitative. This was based on the fact that a use case had to be built in order to test the platform in the context of a specific use case and verify its functionalities as well as explore the benefits of adopting a choreography-based approach to implement an ITS application. The research was conducted with an exploratory and applied approach, which is applicable when a theoretical analysis is not adequate enough. It was built upon existing research that was at that time being done regarding the under-development platform and the way the researchers developing this platform were trying to provide a more efficient and cost-effective solution to already existing problems. Figure 2.1 provides an overview of the research process.

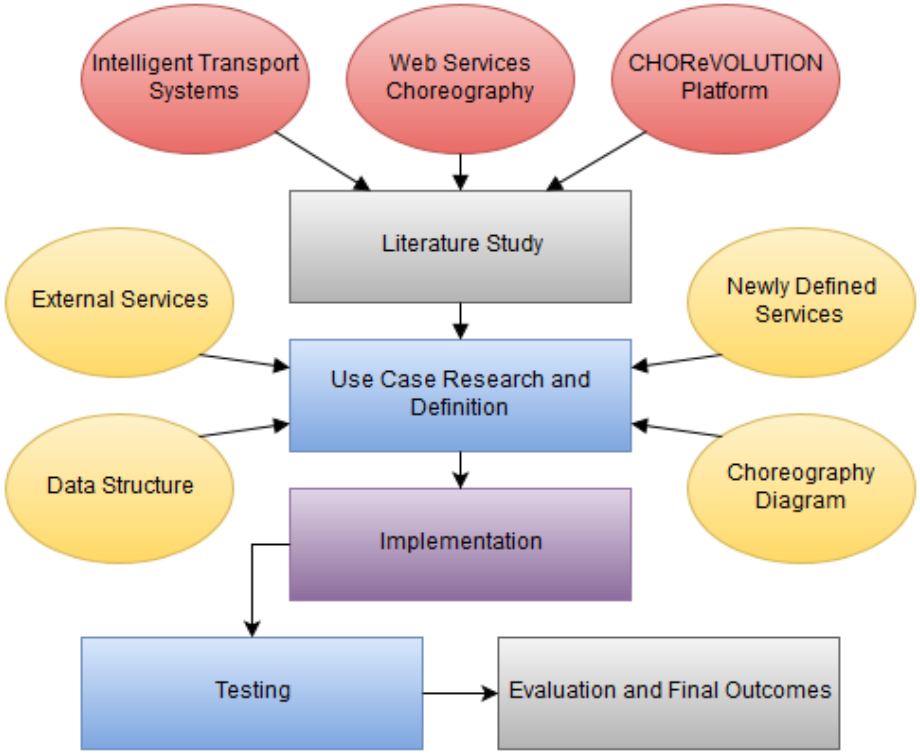


Figure 2.1: Overview of the research process

The below sections describe, in a more detailed way, the research methodology followed and how the research was conducted throughout the thesis project.

2.1 Literature Study

The literature study and review was the initial phase for the research conducted and is presented in Chapter 3. The ITS domain is wide and can be divided into several sub-domains [9]. It was essential to understand the requirements of different ITS applications and the state-of-the-art methods for fast prototyping of traffic related applications. By exploring this area and more specifically the area of Advanced Driving Assistance Systems, it was feasible to break it down into smaller and more specific ones and finally focus on the ITS sub-category that can be benefited from the choreography approach, as well as have social and business potential. This was accomplished by identifying sources describing the ITS domain, its history and its evolvement as well as by searching databases that contain major projects concerning this domain and trying to understand which projects have a huge impact on the ITS domain and are worth mentioning.

The next step was to review literature regarding the different methods or approaches to design web services and achieve collaboration between them, e.g., service composition. This was done by extensively searching different databases, like the IEEE database, identifying different sources that discuss the same methods under different points of view and critically evaluating those sources. The purpose of this procedure was to deeply understand the different and most dominant approaches for service composition, and investigate what a choreography-based approach can offer compared to other methods that exist for similar purposes. This helped to motivate the usage of choreography-based approaches within the ITS domain and potentially give insights on other similar application areas. During the whole process, it was important to have in mind that the domain of interest is the ITS domain and to find, if possible, any previous work regarding the web services choreography and this domain or compare it to other domains where this approach is used. It was also crucial to check if there is any principle, outcome or approach that could be applied to the ITS domain too.

Finally, the last step was to explore the platform and gain better understanding on its structure and functionality, the different components that it includes, its capability and how it could serve the ITS use case that is under

investigation. This was done by studying and reviewing the deliverables of the project under which the platform was developed as well as papers submitted in conferences containing outcomes of the project. It should be kept in mind that the platform development was ongoing at the time this thesis was conducted and thus not all features and components were available. This also was an opportunity to learn the whole research and development process of such a platform.

2.2 Use Case Research and Definition

After having a clear view of the ITS domain, the web service choreography approach and the platform, the research focused on designing and demonstrating a concrete use case. A number of decisions had to be taken carefully, such as what the use case should do in detail and what should the business goal of the choreography be.

Once these questions were answered, an extensive research had to be conducted regarding the web services and their methods that needed to be defined and created, as well as what external services and APIs could be leveraged and how all these services should interact together in order to build and realize the choreography. More specifically, research on already existing external APIs and data sources was very essential to find out what information could be extracted and how this information could be used by the use case and benefit it. The term external APIs refers to third-party web services and APIs that are already implemented and would provide real time data to the use case and the newly defined web services.

All this research and the information extracted had to be aggregated in order to design a concrete use case that would be expressed through a choreography diagram. Moreover, a data structure had to be defined to organize the data that would be created, gathered, processed and presented. The whole design had to be simple to understand and easily expandable for future work purposes, as this was the first use case ever to be created and tested for the platform.

2.3 Choreography Implementation

This step included the exploitation of the selected external services and APIs, the deployment of new web services, the implementation of the use case and the integration of the platform. The procedure involved further exploration of

the platform and its components in order to achieve a functioning application that is based on a choreography approach and is able to be deployed over the platform.

For the implementation, the iterative and incremental development method [10] was chosen. The basic reason for this choice is that it allows development through repeated cycles or iterations and in smaller steps at a time. This method is highly suitable in this case as at each iteration it is able to take advantage of what was learned during previous iterations. This has the benefit that the actual learning and experience came not only from the development of the use case but also from the gradual use of the platform, which was at that time under development and many changes were made in terms of implementation, making the use of iterations more than essential. The following figure visualizes the implementation process as well as the steps that were followed during each iteration.

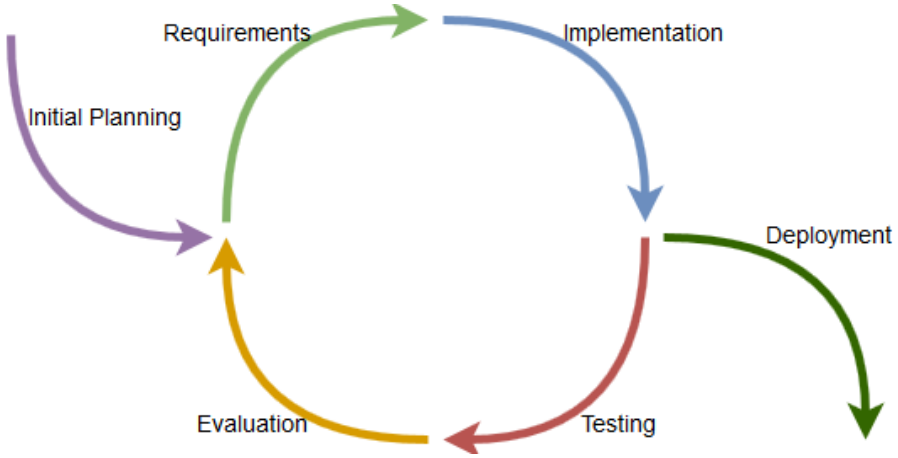


Figure 2.2: The different phases of implementation

It should be mentioned that in the above figure the term “Evaluation” does not imply evaluation of the use case or the platform but only evaluation of the functionality of what was implemented during that specific iteration.

2.4 Choreography Experimentation and Evaluation

In order to test the use case that was implemented and answer the scientific questions defined on section 1.3, a user application had to be designed to feed the necessary input into the choreography and execute it. A series of tests had to be designed to specifically answer each question.

In order to answer question 1, the use case had to be deployed over the platform and check if the platform is able to automatically produce the required components. By confirming that the deployment process is completed without any warning or exception from the platform and the choreography is ready to be executed, it would be able to safely state that the platform is able to produce the required components for this specific use case. In any case of deployment failure, then the opposite would be proved. It has to be mentioned that the ability of the generated components to accurately coordinate the interactions between the services was not something to be answered by the procedure described above.

In order to answer this, which is in essence the second question defined in section 1.3, the interactions between the services had to be tested and evaluated. This was more than necessary to make sure and verify by observation that for the specific use case, the platform and its components are able to coordinate the services and that no undesired interactions between the services occur. This had to be done by running multiple tests giving different random inputs from the user application and monitoring the interactions between the services and the sequence of the methods' invocations. This sequence had to be logged for each test in an external file and be compared to what is defined from the choreography diagram. Actually, it should be fairly easy for the use case developer, since he is the one who implemented the choreography diagram and the services, to know the exact expected sequence of the methods' invocations. In essence, correct sequence of invocations would mean that the services coordination was successful and as a result question 2 would answered. The answer to this question does not serve as a proof of the correctness of the platform but rather as a first step to showcase the capability of the platform and its components with different inputs under the constrains defined under question 2 in section 1.3.

The answer to question 3 could be reached by combining the experience gained and the observations and explorations made from the study of the platform, the design of the use case and the implementation of the services. As a result, it was easy to identify, understand and list these factors. Additionally, once the factors were listed it was not difficult to understand which ones could be controlled and affected by the developer of the use case in order to decrease the latency inserted by them and improve the overall latency of the choreography. Thus, question 4 could be answered.

Moreover, what had to be measured was the time required to execute and complete the choreography. This had to be done in order to answer question 5 and find out information regarding the latency of the overall system. A number of tests

had to run by submitting different random input from the user application, which would result in different amount of data to be processed and passed from one service to the other. During each test, a timer, set in the code, would calculate the time that took to complete the test. This time along with the amount of the different data that had to be processed, would be gathered, presented and evaluated in order to check how much the choreography execution time may vary.

It has to be mentioned that from a user experience perspective, having a large execution time variability and high execution times may disappoint the end users and turn them away from the user application. While it is out of scope for this thesis, the thesis aims to provide statistical results for application designers and developers and it is up to them to determine what the users believe to be an acceptable response time.

Finally, the last question, which is question 6, had to be answered. The way to identify the main factors that contribute to the variability of the choreography's execution time, was to take a closer look at the results of the experiments that would be performed to answer question 5. This could greatly help to identify the possible reasons or factors that cause this behavior. What had to be done was to isolate each factor and run tests to confirm that these were indeed the main factors that contribute in the variability of the execution time. For example, if from the tests conducted to answer question 5, two reasons, A and B, were suspected to mainly be responsible for the variability, then three series of tests would have to be conducted. During the first test series, the latency would have to be checked when these factors were not present during the choreography execution. If the latency variability from these tests was much lower than the one observed during the tests for question 5, then it would be clear that the right reasons for the high execution times had been identified. During the second series of tests, factor A would have to be present while factor B would have to be absent. In this way, it would be possible to analyze, measure and report how factor A contributes to the variability of the execution time. The same tests would have to be conducted where factor B would be present and factor A absent in order to understand the contribution of factor B.

The way each factor had to be isolated or excluded from the tests depended on the nature of the factor and was not possible to be absolutely defined before the factor was identified. However, two possible methods or ways to exclude a factor would be either by submitting specific input to the choreography or by modifying the executed code to prevent the appearance of this factor. Hence, the

exact way to prevent a factor from being present during the tests, had to be determined once the possible factors were pinpointed.

2.5 Conclusion, Final Outcomes and Future Work

During this phase, by using the implemented use case and the results of the conducted tests as a starting point, the scientific questions presented earlier were answered. Moreover, it was discussed what the potentials of the platform are in general as well as more specifically when it comes to the ITS domain combined with concepts like the Future Internet, Smart Cities and the Internet of Things. What is more, the limitations of the current work were identified and mentioned. Keeping in mind that the work regarding the platform development was during this thesis still in progress, it was also explained how the implemented use case could be further expanded in order to take full advantage of what the platform aims to offer and leverage the concepts mentioned above.

3. Intelligent Transportation and the CHOReVOLUTION Platform

The benefits of the development of Intelligent Transportation Systems (ITS) are multiple. However, the advancements made on this field are tightly coupled to advancements made on other fields. For example, in order to build efficient ITS web or mobile applications, innovative methods and tools should be used regarding the architecture of the web services that should be deployed. This fact makes the deployment of ITS over the proposed platform a very interesting and promising idea to be explored.

3.1 Intelligent Transportation Systems

Intelligent transportation systems are important to address traffic challenges and form one of the components of smart cities. Since traffic jams are a major social problem which is growing more and more and the expansion of the current transport network is not always possible, especially in dense urban areas, ITS are needed more than ever before to improve the efficiency of the existing transport network and reduce congestion. The combination of a better transportation infrastructure and state-of-the-art IT technology can benefit vehicles, drivers and the transport network with efficient methods to improve the control of the transportation and reduce traffic in a safer and more convenient way.

By using ITS applications, it is greatly expected that congestion in dense traffic areas will be reduced [11]. This can be achieved by advising drivers with real-time information regarding traffic situations and possible alternative routes, and allow them to avoid areas with high congestion. Moreover, drivers' comfort will be improved since less time will be spent in traffic jams and thus travel time will be decreased too. In addition, it has been argued that ITS have a beneficial impact on the environment as they promote eco-aware driving. More reliable and accurate routing information has a positive impact on the travel time, reducing atmosphere pollution and fuel emissions. Safety benefits could include reduced number of accidents and property damage. Furthermore, reducing the number of accidents would keep the road lanes open, which would otherwise lead to further congestion and situations that may contribute to further accidents and damage.

For this reason, ITS have been a major research over the last decades and ITS solutions have been studied, developed and deployed all over the world. The development and evolution of ITS depends on many other fields like electronics,

communications, information systems and signal processing [9]. The most significant attempts have been made in Europe, USA and Japan [12]. There are several leading projects from each country/region that could be mentioned. The Connected Vehicle Research (CVR) project in USA [13] aims on improving the connected vehicle technology and develop state of the art connected vehicle applications. Another research program, eSafety [14], was funded by the European Union in order to improve road safety by developing vehicle-based intelligent systems, while the Smartway program [15] in Japan was aiming to build a cooperative intelligent transport system where traffic information, collected by sensors, was analyzed and sent directly to car drivers.

3.2 Advanced Driver Assistance Systems

Advanced Driver Assistance Systems (ADAS) are part of the ITS development. The term ADAS covers a broad range of systems, from systems that provide information and warnings to the drivers to systems that provide fully automated driving [16]. Thus, ADAS could be divided into four main categories [17]. The first category is systems that provide fully automated driving where the driver has no active role and cannot intervene to the decisions made by the system. The second category includes systems that intervene to the vehicle control but still the driver has an active role. There are also systems that are designed to provide feedback and warnings to the driver according to his driving behavior in order to reduce his errors and prevent him from violating the Highway Code. For example, there are systems that can assist the driver when performing a lane change or can help him avoid collisions. The last category includes systems that their purpose is simply to provide helpful information to the driver. This might include navigation services and information regarding congestion, roadworks or other road conditions. This type of systems is also known as In-Vehicle Information Systems (IVIS).

IVIS can further be divided into several categories [18]. The first one is In-Vehicle Routing and Navigation Systems that provide different services for routing from one place to another destination. Such services might be trip planning which allows the driver to plan his trip and select a specific route, dynamic route selection where the route is determined based on current traffic situation and other related info and can be recalculated if needed and finally route guidance, which for example may provide turn-by-turn guidance by using voice commands or icons.

The second category is In-Vehicle Motorist Service Information Systems which are able to provide broadcast information on commercial services that a driver passes by or a database which can be accessed by the driver in order to retrieve information about these services, for example about local attractions.

The last type of IVIS are In-Vehicle Signing Information Systems that can provide various information about warnings and road condition as well as advisory information. This could include advice regarding the speed limit, traffic lights, stop signs or sharp road curves. Until now, the In-Vehicle Routing and Navigation Systems are the most commonly implemented IVIS systems.

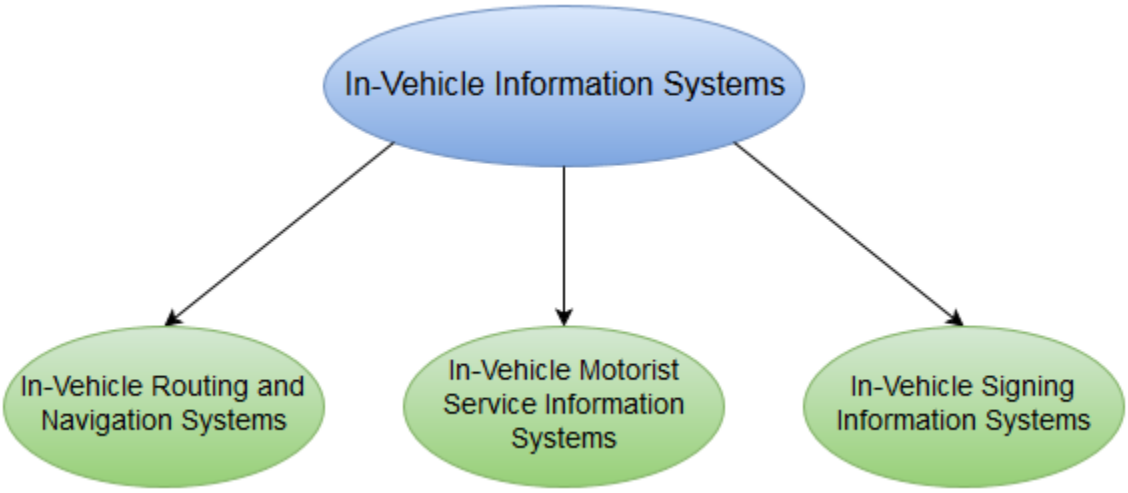


Figure 3.1: Categories of In-Vehicle Information Systems

However, in order to build such systems and deliver their benefits to the end user, the appropriate web services should be deployed. The purpose of these services is to gather relative data regarding congestion, road conditions or any other necessary information, evaluate and process this data and deliver it to the end user. For this reason, efficient and innovative ways and platforms to build the necessary web services should be deployed.

3.3 Web Service Composition

Web services are modular, self-describing and Web accessible distributed software components. They can be published on the Web and be discovered by software agents. Web services are becoming more and more important for the implementation of web applications. However, the challenge that needs to be

addressed is how to combine all these services in order to build a more complex business process. This procedure is called Web Service Composition (WSC) [19].

There are two main ways or architectures to compose different web services. One possible way to combine different web services is the Web Service Orchestration. By utilizing the orchestration model approach, the web services do not enact together from a global perspective. There is always another central process, it may be another web service, which serves as a central controller and is responsible to coordinate all the involved services [20]. In that sense, all the web services do not know that they are participating in a higher level business process or application as all of their methods are invoked exclusively by the central controller. The Web Service Orchestration is described by using an orchestration language so that the interactions between the services can be described [21]. Currently, the most widely used language to describe these interactions is the Business Process Execution Language (BPEL) [22]. Both the industry and the open-source communities have developed a number of BPEL engines, like the IBM Websphere Process Server [23] and the OW2 Orchestra [24].

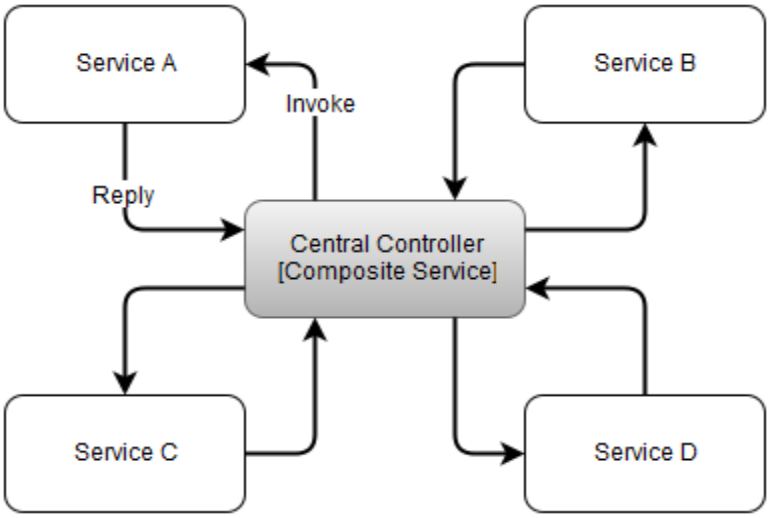


Figure 3.2: Web service orchestration paradigm

Web Service Orchestration is not the only way to compose web services. There is another architecture called Web Service Choreography. Choreography is a distributed approach to compose web services within which web services are called participants. It allows multi-party services from multiple domains to collaborate with no centralized coordination. The choreography defines all the possible interactions between the services in a peer-to-peer fashion [25]. All the message exchanges between each participant do not require the presence of a centralized coordinator. In fact, all these message exchanges have to be defined

from a global point of view [26]. In contrast to the orchestration model, every web service that takes part into the choreography should be fully aware of how and when to interact with the other services. In this sense, each choreography is a collaboration between different services in order to realize a higher-level business process. All the participating services should know what operations to execute, the messages that need to be exchanged and when this message exchange has to take place. A choreography model can be described by using a language such as the BPMN2 [27], the Web Service Choreography Interface (WSCI) [28] and the Web Services Choreography Description Language (WS-CDL) [29].

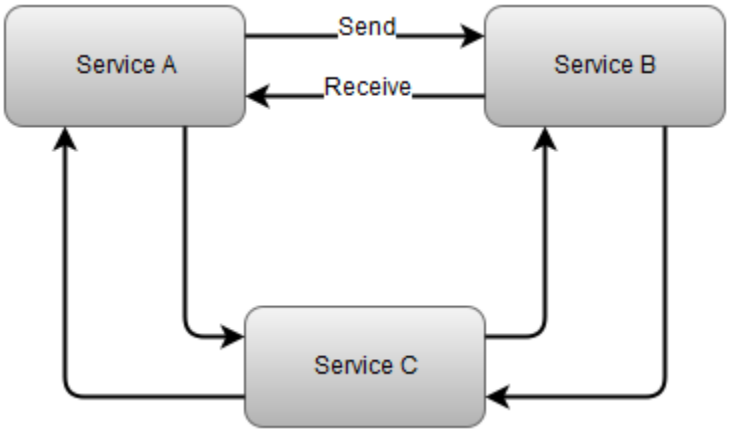


Figure 3.3: Web service choreography paradigm

There has been a great argue regarding which of the two different approaches is more reliable, robust, scalable and efficient. Each of the two approaches have their advantages over the other. While orchestration is more suitable for single-domain applications where all services can be controlled, the web service choreography approach offers some benefits that cannot be ignored when it comes to large-scale web services and applications involving multiple domains. First of all, choreographies are more efficient when taking into account the amount of data that has to be computed and exchanged [30]. When using the orchestration approach, all the data has to go through the central controller, since it is the single point of contact and responsible to invoke all the services. Thus, transferring large amounts of data to the central coordinator may result in overload, bandwidth waste and performance degradation [20], [31]. From this point of view, the Choreography approach seems to perform better in terms of efficiency and scalability. What is more, the controller itself can be a single point of failure which may cause the whole system to collapse since the whole business logic is only in one process or service [32]. This does not apply to web service choreographies, where the absence of a central coordinator enhances the robustness of the higher-

level business process. This means that the business logic is not concentrated in a single service or process but is distributed among different processes. In case a process fails and depending on the choreography architecture, the failure might not affect the execution of the whole choreography but only a part of it. Choreography is also a fairer model since every service acts as a peer or participant and there is no central control by another process. Last but not least, the fact that all services are peers and have to collaborate together is helpful to avoid deadlocks, which may occur when each peer executes its own orchestration plan [33].

It has to be mentioned that until now the most dominant approach for composing web services in general and more specifically when it comes to the ITS domain is the orchestration approach. A large number of research work that focuses in orchestration and the ITS domain and proposes different orchestration approaches for the development of ITS web services can be found in the literature such as in [34], [35] and [36]. However, no work or research was found regarding the use of the web service choreography for the development of ITS services and applications. Sources and databases investigated for this reason include all papers and work submitted in all the ITS World Congresses since 2000, the Vehicle and Road Automation [37] which is a support action funded by the European Union and includes an extensive catalogue of projects related to this field, the knowledge center of ITS America [38] and ITS Asia Pacific [39] as well as the ERTICO – ITS Europe [40] and the IEEE digital library [41].

Most of the research found that is trying to correlate the web services choreography to a specific domain is focusing on e-Business, e-Commerce and e-Governance. Examples of such works can be found in [42], [43], [44] and [45] and is only a small portion of proposals and work done to apply the web services choreography approach to these domains. Some of them propose a pure choreography approach while others examine hybrid models where both the choreography and the orchestration approaches are used. Another domain where there was found some research within the web service choreography is the bioinformatics domain. Examples of trying to use the choreography approach for the processing of bioinformatics data and the control of the workflow in this domain can be found in [46], [47] and [48]. Most of the work found, such as in [46], focuses on the increasing amount of bioinformatics data that has to be processed which creates a bottleneck in the central controller, a fact that constitutes choreography an appealing alternative option. Other works, like [48], focus on the creation of a hybrid model combining choreography with orchestration to manage large amounts of data.

One could argue that approaches from the aforementioned fields could easily fit into the ITS domain since in the end it is all about web services. However, it should be considered that services within the ITS domain have special requirements. These services have to deal with real time data that may change within milliseconds, especially when it comes to safety-critical applications, are time sensitive, since decisions may have to be taken within milliseconds and require high demand of accuracy. All these are unique requirements which are not taken into consideration in the work found in the other domains since the nature of the services and the requirements there are different. For example, in the case of bioinformatics the requirements were mainly related to the huge amount of data that has to be processed and the management of many large databases. On the other hand, for the domains of e-Business, e-Commerce and e-Governance there is a lot of focus on security related requirements, such as in [42] and [49].

3.4 Platform Overview

The CHOReVOLUTION platform is an effort to leverage all the above-mentioned advantages of the chorography web service composition approach in order to automatically synthesize large scale, dynamic and secure choreographies. This project is based on work previously done on the CHOReOS project [50]. The motivation behind this project is the challenges of the Future Internet. These challenges [51] can be categorized in technical, like the development of innovative software applications and services and non-technical such as socio-economics, business and environmental issues [52]. In the next years, more and more devices such as smartphones, tablets, cars or any other devices that can be benefited by the Internet are already or will be connected. This fact is a great opportunity for the development of new services and systems that could simplify many social and business sectors. The next step is to combine all these systems and services together and address a number of issues such as coordination, scalability and data heterogeneity.

Taking into account these new challenges, during the CHOReOS project, an integrated development and runtime environment (IDRE) was implemented in order to enable the composition of complex choreographies within the Future Internet and the development of large scale applications, assisting the developer during the software life-cycle, from the specification of the requirements, to the design of the choreography as well as its synthesis and enactment [53]. One strong conclusion from this project was that the CHOReOS IDRE could be further

developed so that it can be adopted from the industry by enabling the synthesizing of evolvable, adaptable and secure choreographies.

The purpose of the CHOReVOLUTION project is to address the outcomes of the above conclusion. Choreographies may be in operation for a long time. During this time some changes might occur and the platform should have the necessary mechanisms to support these changes. Two different types of changes can be identified. The first one is goal changes in the sense that as business procedures evolve, it is possible that changes might have to be done in the initial choreography specification. The second is changes in the context of the choreography. This implies that services cannot stay the same; sometimes they need to be substituted or modified according to the needs of a business organization. Since replacing the whole choreography with a new one is not considered to be an optimal solution, the platform should realize choreographies that will be able to evolve and be adaptable to new requirements and technologies.

Figure 3.4 presents a high-level view of the platform. Web service developers and choreography designers collaborate together to define the business logic, identify the services that are needed and implement them and design the choreography by creating a BPMN2 choreography diagram. This diagram is used to express all the possible interactions between the participating services and how they should collaborate together. In the CHOReVOLUTION project, this is done by using the Eclipse BPMN2 Modeler [54]. The services are published to the Service Inventory while the BPMN2 choreography diagram is given as input to the Synthesis Processor, which selects from the Service Inventory all the services that should take part in order to form the choreography. Finally, the Enactment Engine is responsible for deploying the choreography.

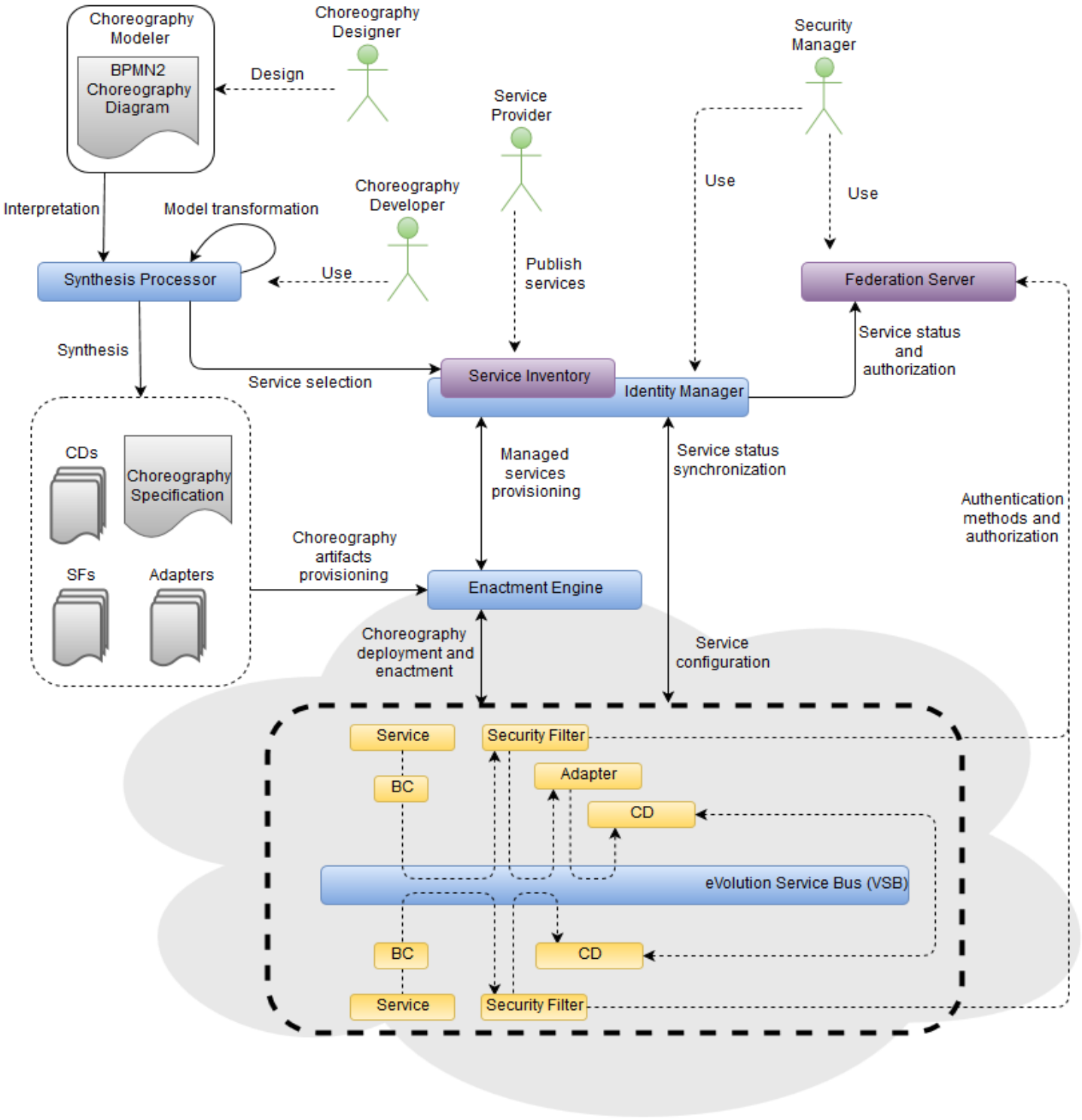


Figure 3.4: CHOReVOLUTION platform overview

3.5 Choreography Diagrams and BPMN2

As mentioned earlier, BPMN2 is a language or modeler used to design a choreography. In general, BPMN2 is primarily used to design and model business processes by providing diagrams. Thus, business procedures can be represented by graphical notations and be communicated in a standard manner. However, BPMN2 is also able to support concepts like choreography and help developers and designers to specify a choreography model through designing a choreography diagram.

A choreography diagram can be described as a set of choreography tasks. Each choreography task is an atomic activity that represents an interaction between two services in the form of message exchanges (request and if applicable a response). BPMN2 choreography diagrams use rounded corner boxes to graphically represent choreography tasks. For each task there are two participants, where the one in the white box is called the initiating participant while the participant in the grey box is called provider. The initiating participant initiates the choreography task by sending a message to the provider. This message is called initiating message and is represented by a white envelope which is attached to the initiating participant. The provider may not or may reply with a return message after receiving the initiating message. This return message, if it exists, is represented by a grey envelope which is attached to the provider. The whole process can be viewed as a token passing procedure, where a token is produced at the start event, it passes through the choreography tasks and eventually is consumed at the end event.

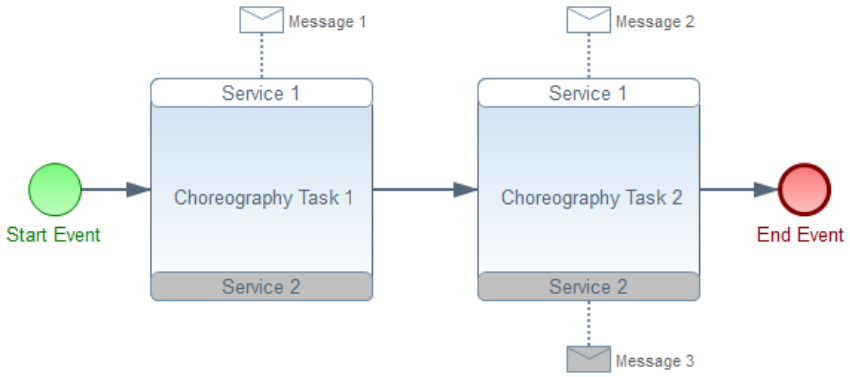
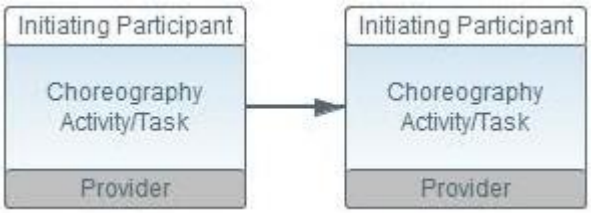


Figure 3.5: Simple BPMN2 choreography diagram involving 2 services

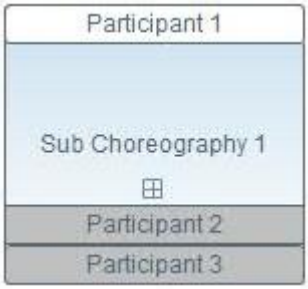
For example, in figure 3.5 the task “Chorography task 1” is followed by the task “Chorography task 2”. In “Choreography task 1”, Service 1 is the initiating participant and sends the initiating message, “Message 1”, to Service 2 in order for the task to be completed. In “Choreography task 2”, which is the following task that has to be executed, Service 1 is again the initiating service and sends the initiating message to Service 2. In response, Service 2, which is the provider, replies back to Service 1 with a return message, “Message 3”, and thus the choreography is completed.

It should be mentioned that the above example is really simplistic and involves only two participants, Service 1 and Service 2. In real world choreographies, the complexity of the choreography diagrams can be high and more elements, like gateways, usually should be used to design the choreography and describe the interactions between the participants. Below, the most common elements and their usage is summarized [55].

Choreography Activity/Task: An atomic activity that represents an interaction that can be one or two message exchanges between two participants.



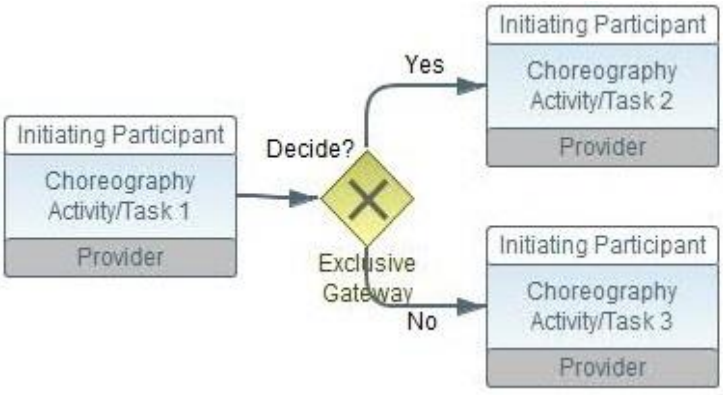
Sequence Flow: An arrow that shows the sequence of the choreography activities/tasks.



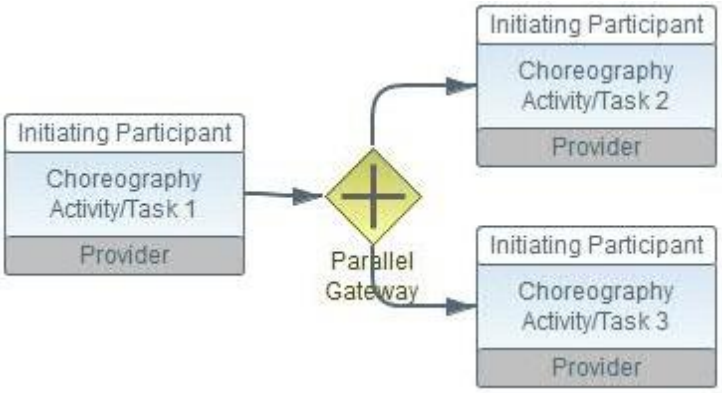
Sub-choreography: A compound activity that is defined as a flow of other activities. Sub-choreographies involve two or more participants and have to be expressed in detail by another separate choreography which describes the interactions between the participants.

Start Event: A graphical marker for the start of a choreography. It can be an event that triggers the choreography.

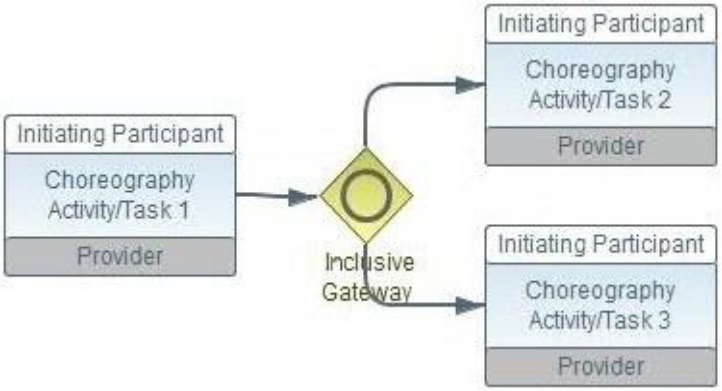
End Event: A graphical marker for the end of the choreography. The choreography is terminated when reaching this marker.



Exclusive Gateway: Creates two or more alternative paths within a choreography. Only one path can be taken.



Parallel Gateway: Creates two or more paths that will be performed at the same time.



Inclusive Gateway: Creates two or more alternative paths within a choreography. More than one path can be taken if the condition for this path is true.

3.6 Choreography Synthesis

The Synthesis Processor is responsible for performing the choreography synthesis, a procedure that produces mainly four software components [56], the Coordination Delegates (CDs), the Adapters, the Binding Components (BCs) and the Security Filters (SFs). CDs are the components that coordinate the interaction between the participating services as defined by the BPMN2 choreography diagram. Adapters are used to bind concrete services to the abstract participants while Security Filters are used to deal with security aspects from both the provider and the consumer side of the service. Moreover, after the synthesis, a Choreography Specification is generated which is used by the Enactment Engine to deploy and perform the requested choreography. It is acknowledged that as of the time of implementing the use case, adapters, BCs and SFs were not yet

available. Thus, the focus is the CDs that are generated by the platform for the coordination logic.

The purpose of the choreography synthesis is to control, coordinate, adapt and secure the interactions between the participating services by using as input the choreography diagram and the services that will participate in the choreography. The right protocol coordination between the services is done by generating the required Coordination Delegates and interposing them between the services. CDs act as proxies to the services and ensure that there will not be any undesired interactions between them. They guarantee the collaboration specified in the choreography diagram through distributed protocol coordination [57].

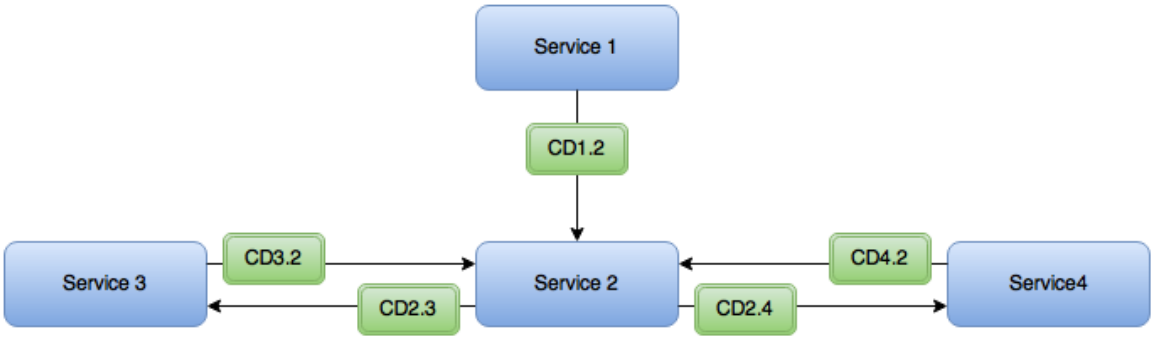


Figure 3.6: A services architecture with Coordination Delegates (CDs)

It should be pointed out that CDs are service independent and always assume that they deal with Simple Object Access Protocol (SOAP) Web Services [58]. However, service interoperability should not be an issue. This is achieved by using the Binding Components (BCs) provided by the platform middleware [59] to interoperate with other type of services such as REST services. BCs are used to achieve interoperability when two different services should interact together, e.g. a SOAP and a REST service, by providing communication protocols primitives.

Since CDs act as proxies for the services, Adapters are generated to arbitrate the interactions between the services and the CDs. They are responsible for solving interoperability issues due to operation names mismatches and I/O data mapping mismatches. For example, Adapters are able to map message data types or merge, split and reorder operation calls and related I/O messages.

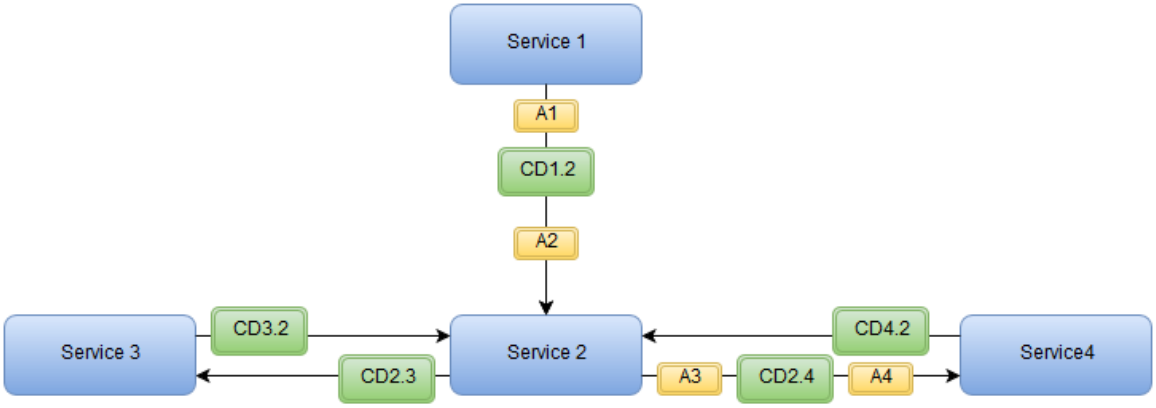


Figure 3.7: A services architecture with CDs and Adapters

SFs are software components that deal with security matters. SFs can be divided into two main categories. Filters for controlling the access to a service and filters which focus on the security configuration and prepare the interaction with the service provider. The security specifications are defined in the service and are used by the Synthesis Processor to generate the SFs. They are interposed between the client service and the CD (or the Adapter if present) and act as reverse proxies working on the request content.

From the consumer point of view, the Security Filter can be used for authorization purposes. It checks the identity of the service consumer and whether the request can be performed. The security information is carried in the request and could be for example a username/password or an X509 certificate. This security information is validated by using the system's Federation Server. Thus, the Security Filters do not decide whether to allow or reject a request to the service provider but they simply enforce the decisions taken by the Federation Server.

From the provider point of view, SFs are responsible for preparing a request before it is sent to the service provider. They can be used to use the client's credentials and request new credentials from the Federation Server that are suitable for the security requirements set from the service provider. The new credentials replace the old ones and the request is sent to the service provider. As a result, the request is changed to adapt to the requirements of the service provider.

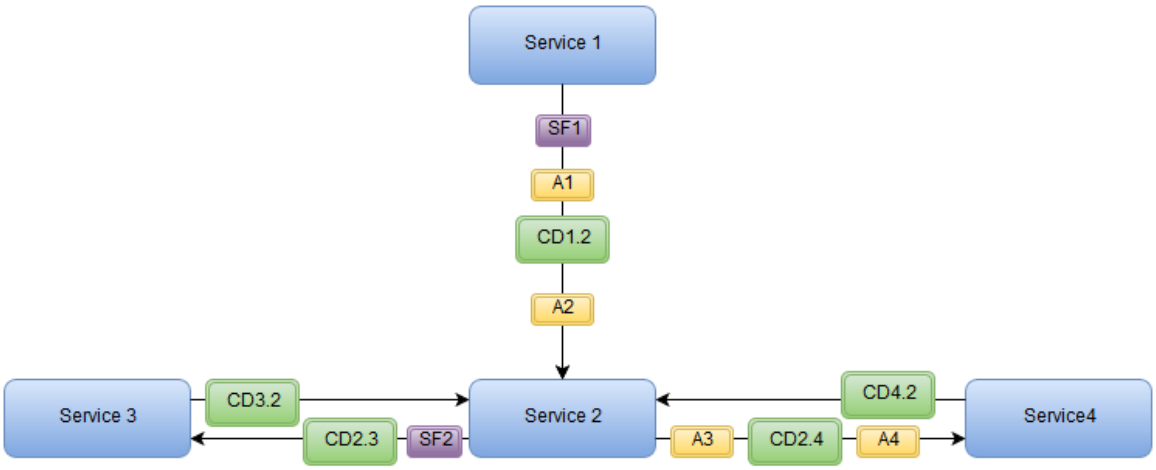


Figure 3.8: A services architecture with CDs, Adapters and Security Filters

4. The Use Case

The first thing that needed to be defined was the exact goal of the use case and the choreography. The aim of the use case was to build a driving assistance application by using the CHOReVOLUTION platform. Given a set of an origin and a destination as input from the user application, web services cooperate together in order to find alternative routes for the specified origin and destination and retrieve traffic information regarding these routes. Finally, these routes are evaluated based on the traffic information gathered for each one of them and the most efficient route is displayed by using the user application.

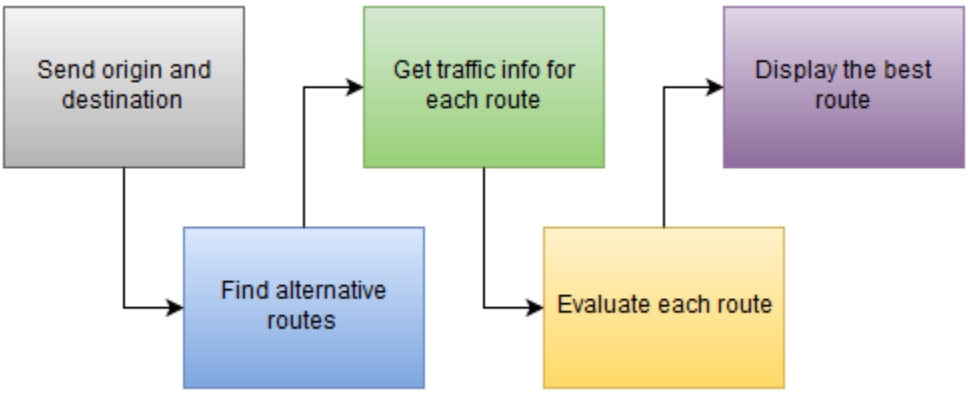


Figure 4.1: *Logical flow of the use case*

As a consequence, the development of the ITS use case for the platform required the selection of external APIs and data sources to provide real time traffic data, the design of the choreography diagram and the web services which would participate in the choreography as well as the definition of the data structure that would be used in order to organize the data.

4.1 External APIs and Data Sources

The development and implementation of Urban Traffic Coordination (UTC) applications depends on various real time traffic data that enable different services. In order to design a choreography that is able to coordinate these services, it is important to identify the available data sources, what kind of data they can provide and how to access and retrieve these data.

For the purpose of this use case, two main kinds of data information had to be retrieved. First of all, it was essential to retrieve a number of alternative routes based on the origin and destination submitted by the user, using the user

application. There are a lot of APIs and data sources that can provide this information. Some of these APIs are the Google Maps Directions API [60], the HERE Routing API developed by Nokia [61], the MapQuest Directions API [62] and the Mapbox Directions API [63]. All of them were evaluated based on the information that could be retrieved and leveraged by this specific use case, the complexity required to retrieve the information and the accuracy of the information that was returned.

What is more, the second type of data that had to be retrieved by external data sources is traffic information based on the retrieved routes. There is a great number of existing APIs that can provide this information, however these APIs are characterized from a great degree of localization. For example, almost no API was found that offers global traffic information and if it was found the information was not 100% up to date. Most of the APIs that are able to provide accurate and real time traffic information cover only a specific country or region. As a result, the Swedish road administration (Trafikverket) API was chosen, since it is able to provide accurate real time data regarding traffic and road conditions in Sweden [64].

To sum up, three external APIs were finally selected. The Swedish road administration API as mentioned previously to get traffic information, as well as the Google Maps Directions API and the Here Routing API to retrieve route directions. The information that can be extracted from these APIs is presented in the following sections.

4.1.1 Swedish Road Administration API

The Swedish road administration provides real time traffic information through their API. The user can make a POST request specifying the kind of data to be retrieved. The data that is returned can be in XML or JSON format.

There are four main types of data available: traffic information, travel time, weather information and static data services. For this use case, only the traffic information was leveraged and used. This includes dynamic traffic information and situations from the road traffic and it contains the following types:

- Accidents: All active accidents
- Road conditions: Regularly updated information about road conditions based on camera images, weather forecasts and information reported by the contractors
- Emergency information services: Information on emergencies and extraordinary events that affect the traffic.

- Road work services: All ongoing roadworks.
- Traffic message services: This information may include unforeseen obstructions and queue warnings.

4.1.2 Google Maps Directions API

The Google Maps Directions API provides directions between two locations, the origin and the destination. It can be accessed by making an HTTP request and the requested information, which in this case is a number of alternative routes, is returned in XML or JSON format, as shown in Figure 4.2.

The returned result holds information such as the origin and destination points, the distance of the whole route and the legs which comprise the route. These legs are smaller parts of the route and represent road segments of the whole route. They have their own starting and ending points, where the ending point of one leg is the starting point of another leg, and if they are merged together they construct the whole route. In the same manner, one leg can be further subdivided into steps.

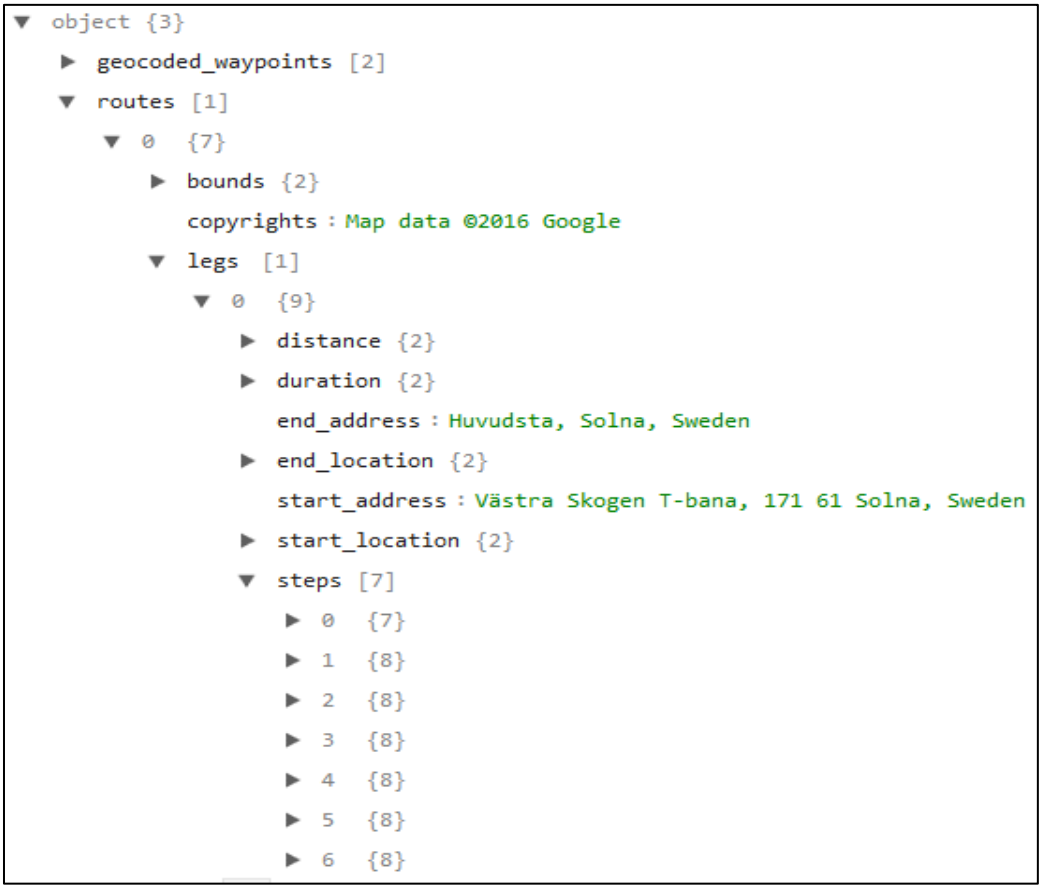


Figure 4.2: *A returned route with one leg and six steps in JSON format*

4.1.3 HERE Routing API

The HERE Routing API works similar to the one developed by Google and is also used to provide routes between two different locations. Information from the API can be requested with an HTTP request and is again returned in XML or JSON format as shown in Figure 4.3.

Similarly to the Google API, the response contains information such as the start and end point, the duration of the travel time and the distance. The route is again organized in legs which are further divided into maneuvers (similar to the Google steps). Below, an example response is shown.

Figure 4.3: *A response containing two routes in JSON format, the first one with one leg and six maneuvers. The second route is omitted.*

4.2 Web Services Design

After specifying the external services and data sources to be used in order to receive route and traffic information, what had to be defined was the new services created from the scratch, their exact role in the choreography, their interactions with the other services as well as the methods each service should have had. In general, the services could be categorized into prosumer and provider services. Prosumer services are the ones that serve as providers and consumers at the same time, while as provider services are considered the services that will be used to get the data from the external APIs and data sources. All services developed are SOAP services.

4.2.1 Provider Services

As implied previously, three provider services were designed, each one responsible for querying one API, getting the response from them and returning the results.

The first provider service, named Dts-Google, is responsible for getting information from the Google Maps API. This service has one method (routeRequest) that accepts as parameters the origin and the destination, which are required for the routes, queries the Google API and returns all the alternative routes retrieved from Google.

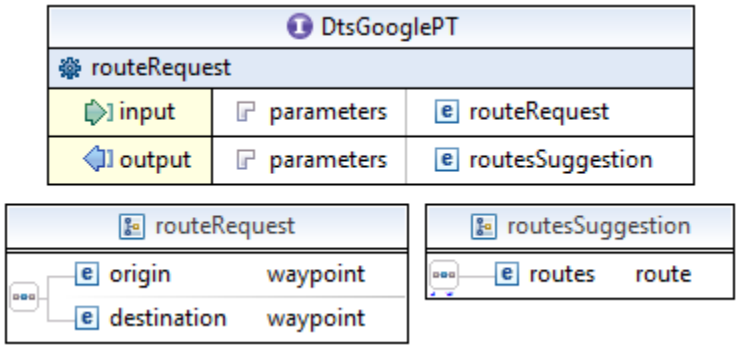


Figure 4.4: *Overview of the Dts-Google service and its method*

The second provider service, named Dts-HERE, is similar to DTS-Google and was created in order to request and receive routes from the HERE Routing API. This service has also one method (routeRequest) that accepts as parameters the origin and the destination, makes an HTTP request to the HERE API and returns the routes that are received.

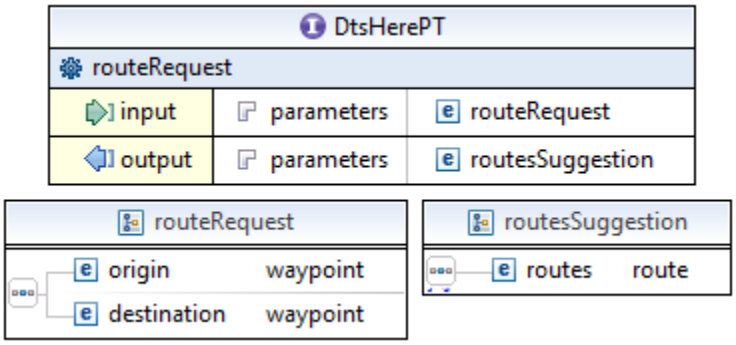


Figure 4.5: *Overview of the Dts-HERE service and its method*

The third and last provider service is named Dts-Trv-Acc. The purpose of this service is to retrieve traffic and accident information from the Trafikverket API for the routes gathered from the previously mentioned provider services. This service has one method (accidentCheck) which accepts as parameter the list of routes gathered from Dts-Google and Dts-HERE and returns the traffic situations and accidents that are found for each one of the routes along with the routes.

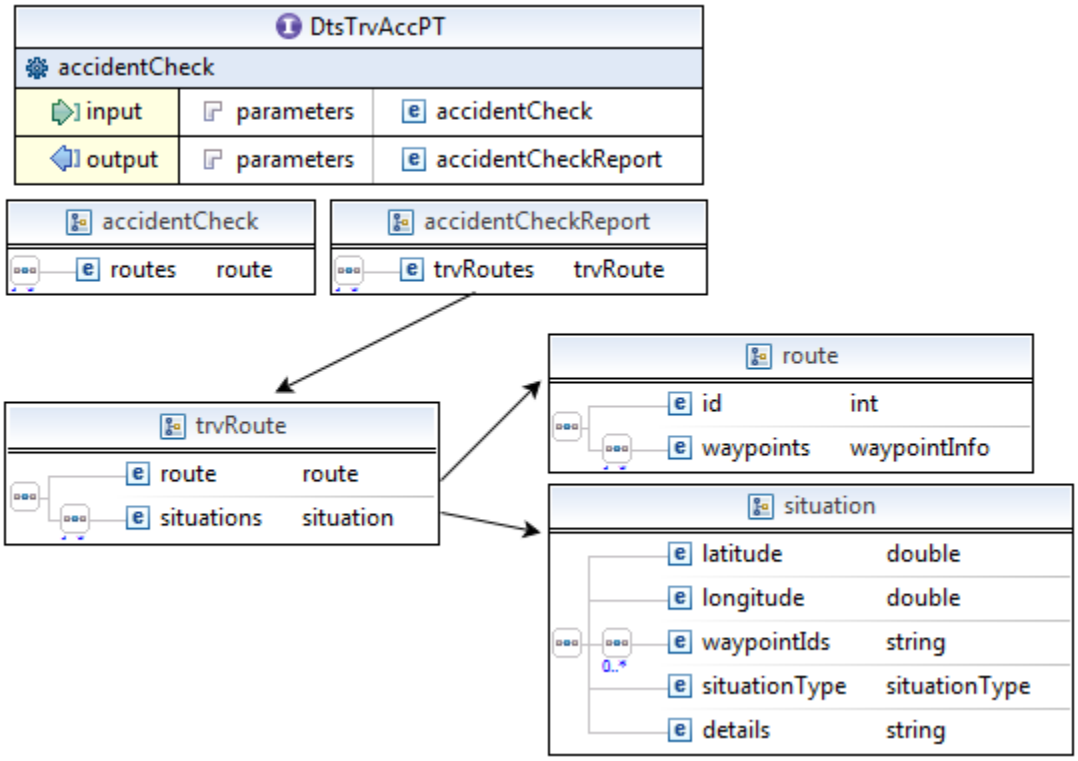


Figure 4.6: *Overview of the Dts-Trv-Acc service and its method*

4.2.2 Prosumer Services

Since the provider services were defined, prosumer services were needed in order to interact with the provider services and return the required data to the user. The prosumer services play the role of the provider for other services and consume the data received from the provider services.

First of all, a service was needed in order to receive the origin and the destination from the end user, send a request for routes to the provider services Dts-Google and Dts-HERE, as well as receive these routes and request a traffic and accident report for all the routes. The responsible service to do this is called Bs-Map. Bs-Map has the following eight methods:

- receiveOriginAndDestination: Method used to receive the origin and the destination from the user.

- sendRouteRequestGoogle: Method used to send a request to Dts-Google in order to find routes for a specific origin and destination.

- receiveRoutesSuggestionGoogle: Method used to receive the routes returned by Dts-Google.

- sendRouteRequestHere: Method used to send a request to Dts-HERE in order to find routes for a specific origin and destination.

- receiveRoutesSuggestionHere: Method used to receive the routes returned by Dts-HERE.

- sendCollectTrvInformation: Method used to send a request in order to collect traffic and accident information for each route.

- receiveReportTrvInformation: Method used to receive traffic and accident information for the routes.

- sendDisplayAndShowSuggestedRoutes: Method used to return the suggested routes along with the traffic information.

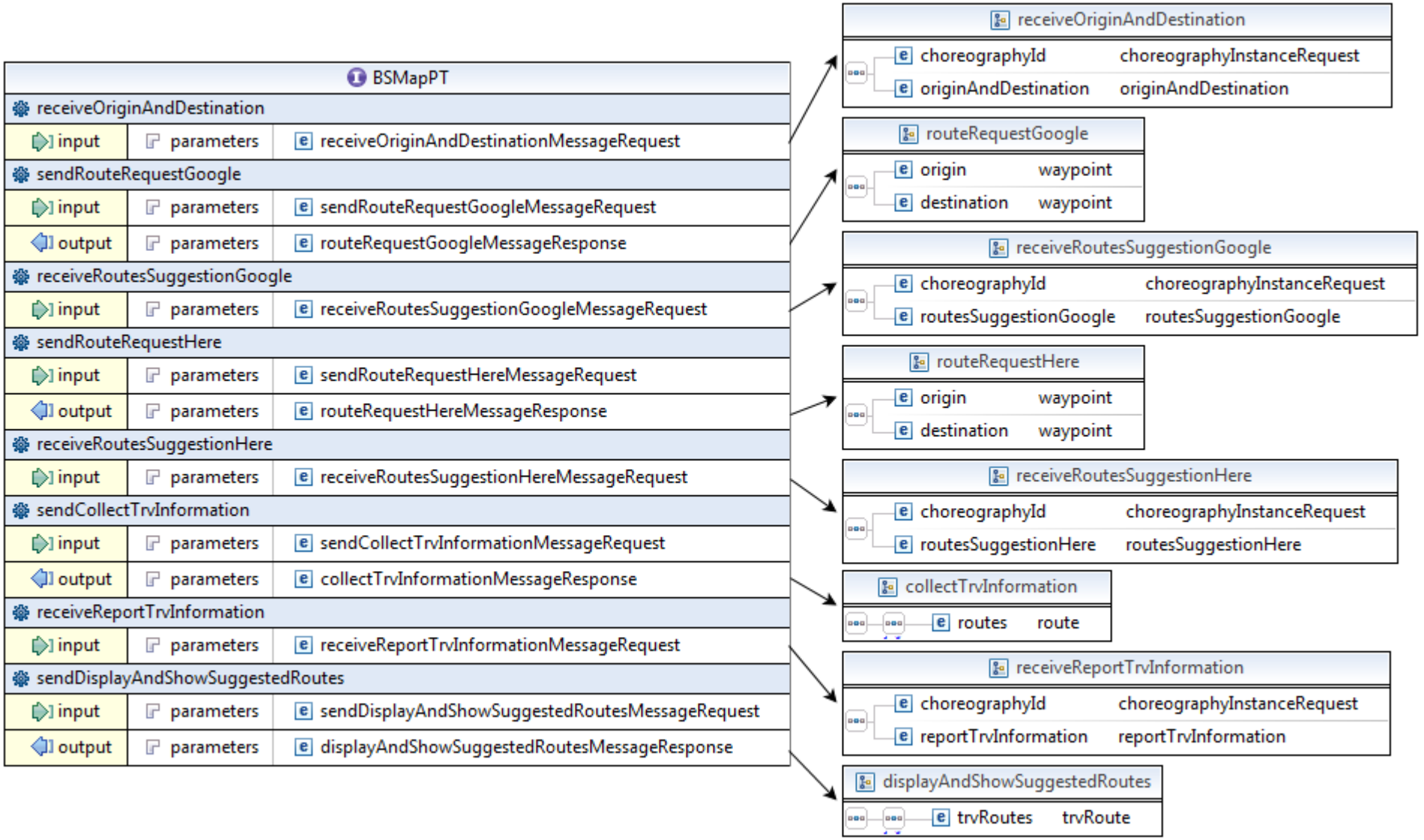


Figure 4.7: *Overview of the Bs-Map service and its methods*

The second prosumer service that was designed and implemented is called Trvc. The purpose of this service is to receive a request for traffic and situation information for the routes and forward this request to Dts-Trv-Acc, which is the service responsible for querying the Trafikverket API. The results from Dts-Trv-Acc are then sent back to Trvc and eventually are forwarded to the Bs-Map service. Trvc has the following four methods:

- receiveCollectTrvInformation: Method used to accept a request for traffic and situation information.
- sendReportTrvInformation: Method used to return traffic and situation information.
- sendAccidentCheck: Method to request a traffic information report.
- receiveAccidentCheckReport: Method used to receive a traffic and accident report.

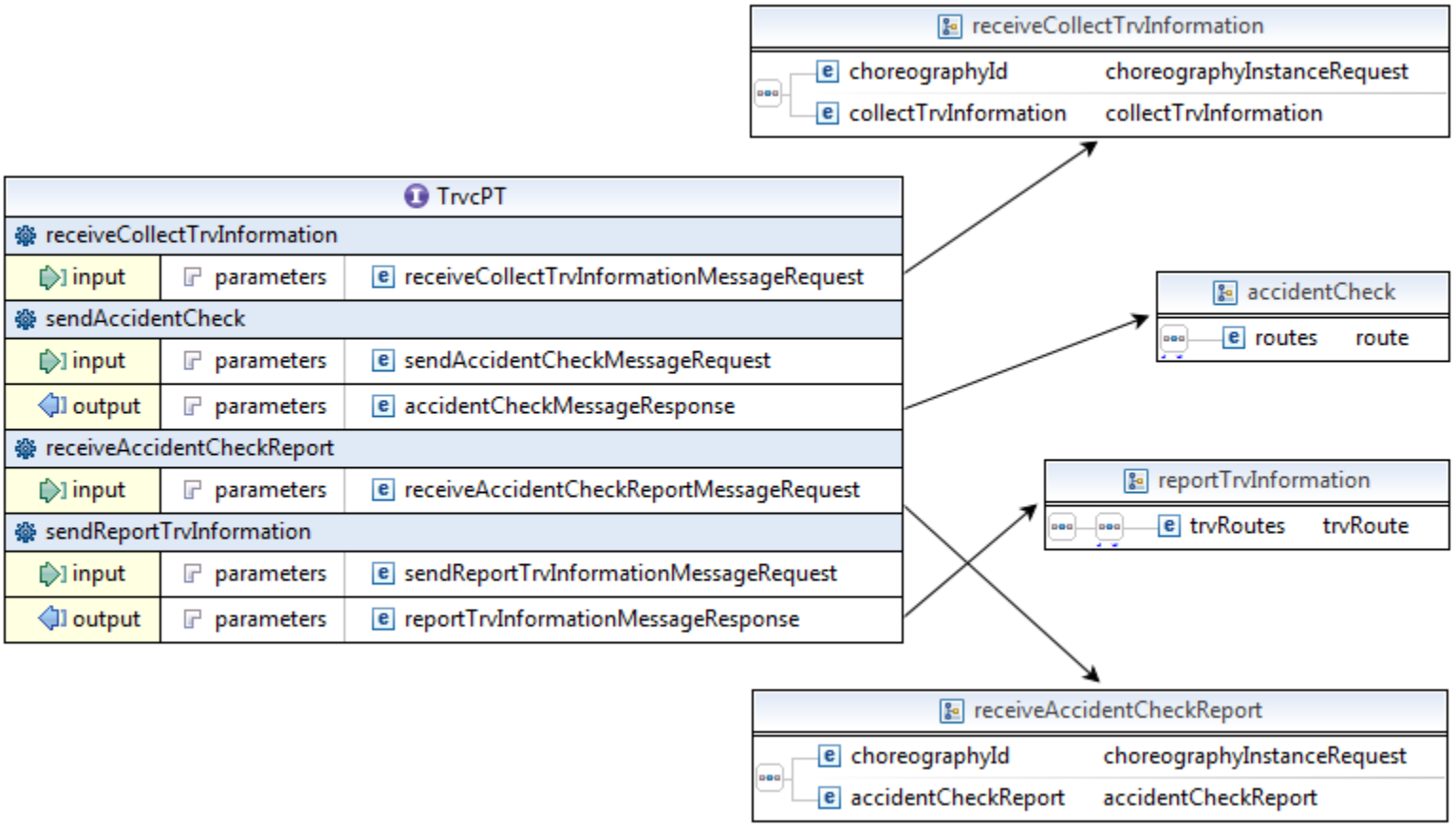


Figure 4.8: *Overview of the Trvc service and its methods*

4.2.3 User Application

The end user web application, called ND, is used to submit the origin and the destination in the form of place names and enables the end user to trigger the choreography. It is also used to graphically display the suggested route on the map along with the traffic situations and information that are found for this specific route. The application is accessible through a web browser and is simple in terms of design.

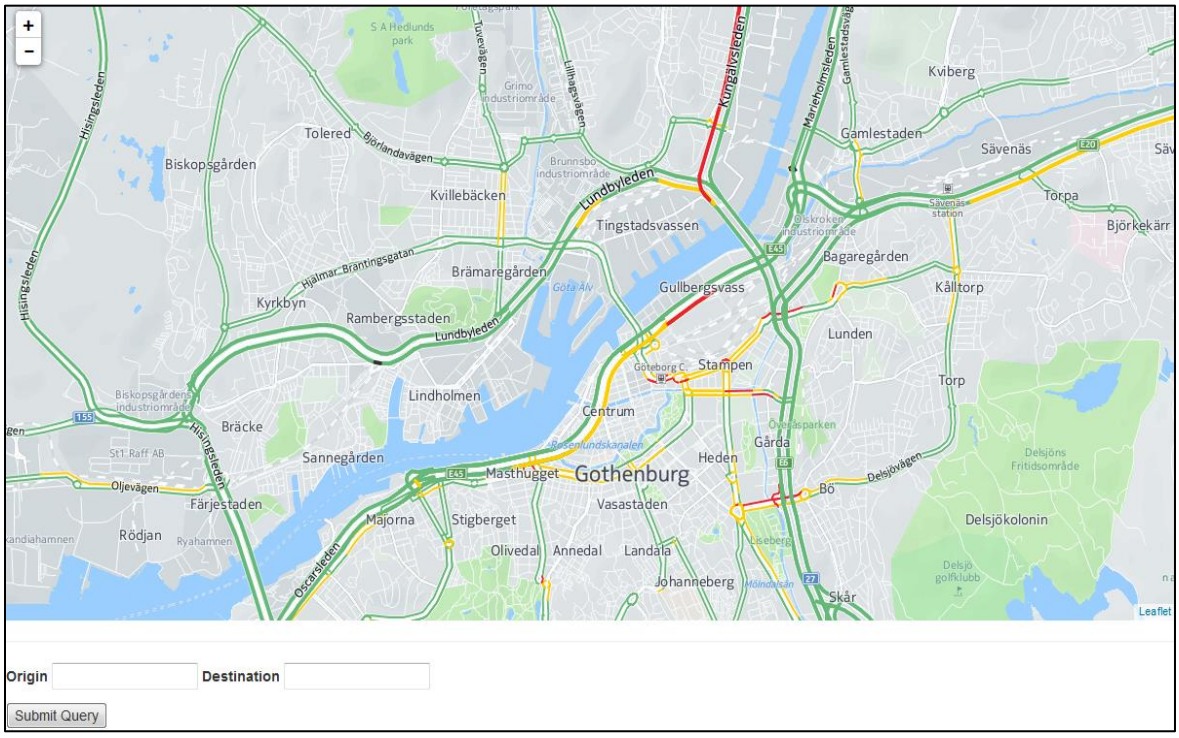


Figure 4.9: *User Interface of the web application*

More specifically, when submitting the origin and the destination, the user in essence makes a POST request to a servlet where these two string values are stored. Then the web application uses the Google Developers Geocoding API [65] to convert the place names into coordinates, a pair of latitude and longitude for each location. This is done because all the other services are designed to handle and process latitude and longitude coordinates rather than place names. When this action is completed, the execution of the choreography should start and in the end the web application should receive one or more routes according to the business logic, if necessary process them and display a route on the map.

4.3 Choreography Diagram

As mentioned earlier, the choreography diagram is taken as input by the platform and its Synthesis Processor. It is the component where all the interactions among the services are defined. Having an accurate choreography diagram is absolutely important as the platform uses it to generate the Coordination Delegates that are used to control the interactions among the services. An inaccurate or wrong diagram would surely lead to undesired interactions with unpredictable results for the choreography execution, where the services will not be able to cooperate together correctly in order to fulfill the business goal.

Figure 4.10 shows the main choreography diagram that was designed for this specific use case, which exploits the services presented in the previous sections. The choreography is triggered by the user who submits his origin and the destination through the end user web application ND. As a result, the first choreography task is executed. The Bs-Map service receives the two parameters and sends in parallel a request for routes to the services Dts-Google and Dts-HERE. When these two services receive the route request message, they query the corresponding API they are responsible for, in order to get routes.

The interaction between Dts-Google and the Google API, as well as between Dts-HERE and the HERE API was implemented internally as part of the business logic of each service and the HTTP request done to retrieve information is part of their source code. Hence, there was no need to output this interaction in the choreography diagram because no coordination was needed between them by the platform.

Both Dts-Google and Dts-Here reply back to the Bs-Map service with a route suggestion message which contains the retrieved routes. When routes from both services are received, a check is performed to determine if the routes received are enough or not. In this specific use case, enough is considered to be one route or more. If not enough routes are received, then the Bs-Map service sends again a request for routes to Dts-Google and Dts-HERE and the same procedure is repeated. However, if the routes returned are enough the Bs-Map appends the routes received from Dts-HERE to those received from Dts-Google and then the Traffic Estimation sub-choreography is triggered.

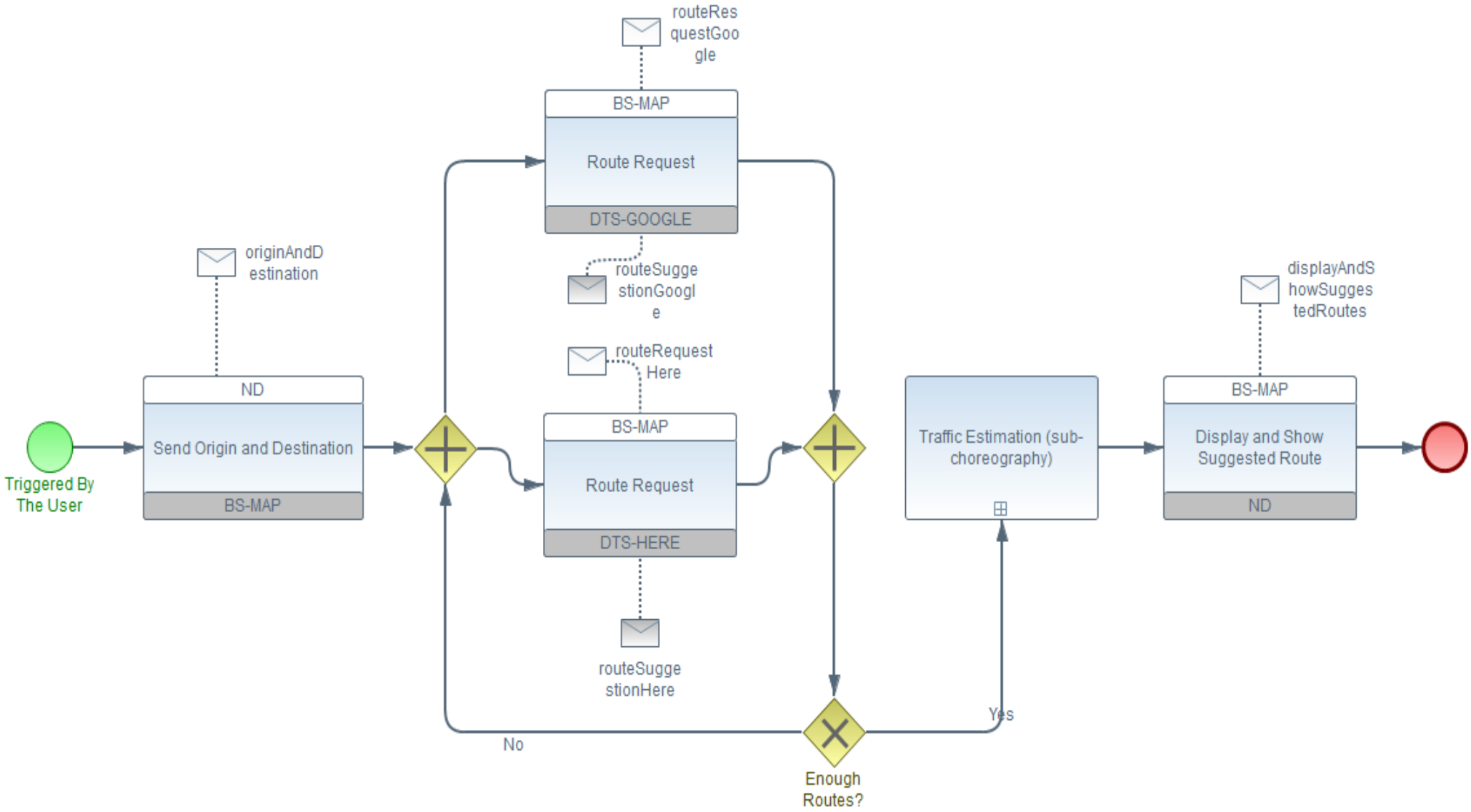


Figure 4.10: *The main choreography diagram*

The Traffic Estimation sub-choreography is presented in figure 4.11. As soon as it is triggered, the Bs-Map service sends a request to Trvc to collect traffic and accident information for the routes that were gathered previously. Trvc sends an accident check request to Dts-Trv-Acc which queries the Trafikverket API for each route. Dts-Trv-Acc sends back to Trvc the report and Trvc sends all the information back to Bs-Map. At this point, the Traffic Estimation sub-choreography is terminated and the flow of the main choreography is followed again. It should be pointed out again that the interaction of Dts-Trv-Acc with the Trafikverket API is not shown in the diagram as the request and the response are handled internally by the code implemented in the service and no coordination is needed by the platform.

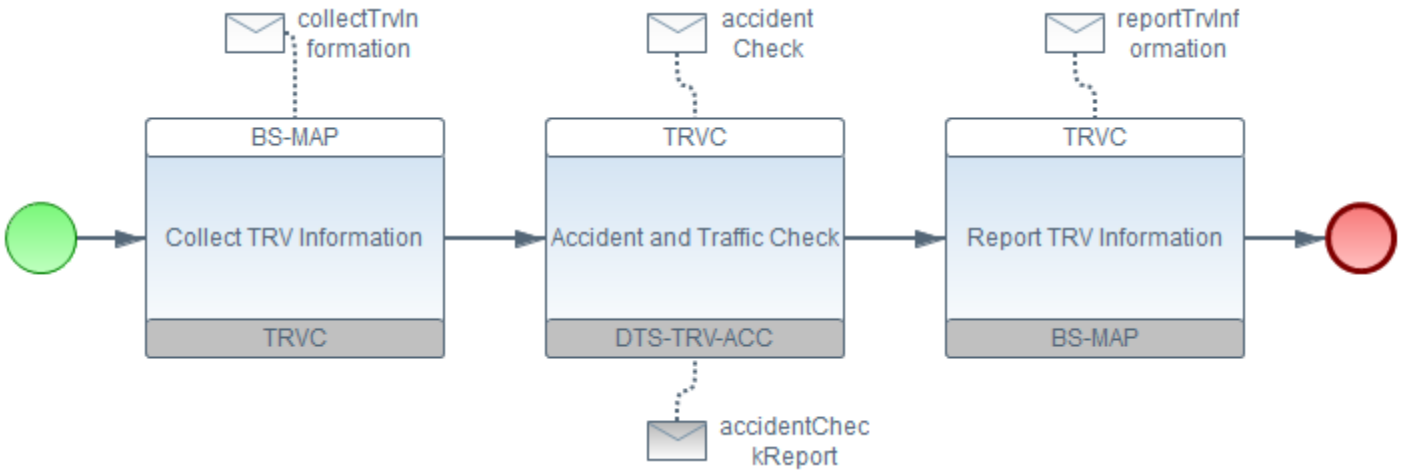


Figure 4.11: *The Traffic Estimation sub-choreography diagram*

The Bs-Map service has now all the necessary information that is needed in order to evaluate the routes. In this specific use case all this information is returned to the user application where the evaluation is taking place. The route with the less traffic situations is selected and presented along with a visual representation of the traffic situations on the route and the choreography is terminated.

The reason that the routes are evaluated in the ND service and not in the Bs-Map is to add more flexibility in the use case in general. For example, by returning all the information to the end user it is possible not to just present to the user the best route as done in this case, but to change this logic, present all the routes and give the end user the possibility to choose the route that he thinks suits him best. It is more convenient to make these changes in the web application after having all the required data rather than in the web services. What is more, as of now, the route evaluation is done in a very simplistic way by choosing the route with the less traffic situations. This is not of course the optimal evaluation that can

be done, however the scope of the thesis was not to come up with an algorithm for route evaluation.

Since there is a clear view of the services created, the external APIs selected and the choreography diagram designed, it should be stressed out again that services with the convention name Dts-* (e.g. Dts-HERE) are all newly defined services created for the purpose of the use case and should not be confused with the external APIs used. Their mission is to query the external APIs and gather the necessary data from them. These APIs are not part of the choreography diagram since the requests to them are done by the Dts-* services internally with an HTTP request and no platform coordination is needed.

4.4 Data Structure

First of all, it should be pointed out that the term data structure or data model that is used here implies the way data is organized and structured in order to be finally presented to the user through the web application. This model had to be flexible, easy to understand and use and expansible in order to include features that might be developed in the future.

Since Leaflet [66] was the chosen tool to create the map in the web application, this greatly influenced the data structure that was chosen. This is because Leaflet provides an easy way to integrate Geographic JavaScript Object Notation (GeoJSON) objects and present the information they hold on the map. GeoJSON is a data structure based on JSON and is used for encoding a huge variety of geographical data structures [67] which makes it very convenient when trying to display a route or traffic situations on the map. A GeoJSON object, which is always a complete GeoJSON data structure, may represent a feature, a geometry or a collection of features. Each feature contains a geometry object with its properties which can be defined by the developer. This fact makes GeoJSON an easily expansible structure in terms of adding new properties. A geometry object might be a point or a line string.

In this use case a point might represent the origin, the destination or a traffic situation and a line string represents a route segment or waypoint. In essence, for each route with traffic information that is returned, two GeoJson structures are created. The first one contains the actual route. It is a feature collection that contains several features. The first two features are two-point geometry objects holding information regarding the origin and the destination while the rest of the features are line string objects holding information for the route segments or waypoints that constitute the whole route, something similar to the legs and maneuvers returned from the Google and HERE APIs.

The second GeoJSON structure that is created holds information regarding the traffic situations for this specific route. It is again a feature collection where each feature has a point geometry object which contains the coordinates of the situation as well as several properties giving more details regarding the situation type, e.g. roadwork or accident. Both of these two structures are easily inputted on the map in order to have the visual representation of the route and the traffic situations.

Figure 4.12 presents, on the left, the GeoJson structure of a route and on the right the GeoJson structure of its traffic situations, visualizing what is explained above. The route structure contains five features. The first one holds information regarding the origin, the second one regarding the destination and the rest of them regarding the route segments or waypoints. The properties can be expanded to include new ones like further route instructions or maybe speed suggestion. The other structure contains two features which are all traffic situations. The properties contain the situation type and a brief description of it.

Figure 4.12: *Visual representation of the GeoJSON structure of a route and its traffic situations*

4.5 Choreography Deployment

Since the services and their methods were defined and implemented, the interactions were defined through designing the choreography diagram and the data structure for the aggregation of the processed data was decided, the last step was to deploy the choreography over the platform. It has to be mentioned that since the platform was still under development at the time of the use case implementation, not all features were available. However, the basic feature that is the Coordination Delegates, which are responsible for the service coordination and interaction, were available and CDs could be generated for the deployed services.

For the purpose of the deployment a retail laptop was used with the following specifications: Windows 10 OS, Intel Core i7-6700HQ Quad Core processor and 16Gb RAM. Two servers were created locally, one server (Apache Tomcat 8) where the services were deployed and another one (Apache ODE) for the CDs that would be generated. Once the services were registered on the local server, by giving as input the choreography diagram into the platform, the Synthesis Processor generated the choreography specification and the CDs. These were passed to the Enactment Engine so that the choreography was deployed and the generated CDs were running on the other local server. It is acknowledged that for the time being, the focus was to experiment and demonstrate the usability of the platform and everything was deployed on a local machine instead of a cloud-based system. While this may not reflect a fully distributed cloud environment, it is believed techniques will be the same when all implementations are moved to the cloud.

In essence, the generated CDs act as proxies between the services and they guarantee the collaboration specified in the choreography diagram. The coordination logic is in them and they are responsible for invoking the necessary services' methods. In contrast to the orchestration architecture, where the whole business logic is in the orchestrator, here this logic is distributed among the CDs. At this point, the choreography was ready to be used by the end users through the ND web application.

Figure 4.13 shows the CDs that were generated and how they were imposed among the services to coordinate and secure their interactions. As shown in the figure, two Coordination Delegates were generated for the services coordination. $CD_{Bs\text{-}Map}$ is responsible for the interactions between Bs-Map and ND, Trvc, Dts-Here and Dts-Google and CD_{Trvc} is responsible for the interactions between the service Trvc with Bs-Map and Dts-Trv-Acc. Of course, the CDs are not visible to the end user or the services developer since they do not have to deal with them when implementing the use case and the web services or when using the web application.

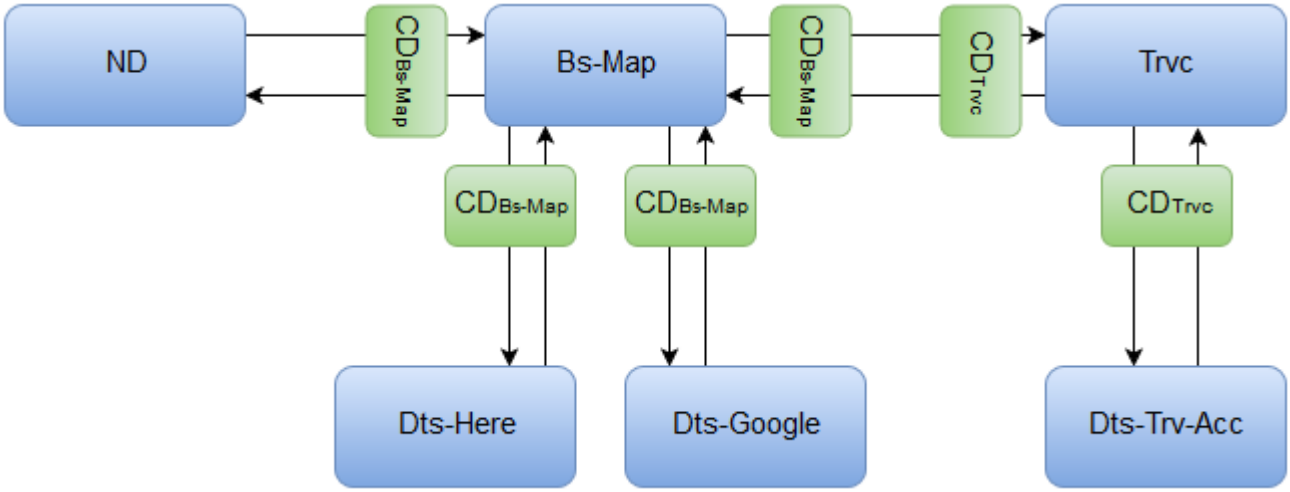


Figure 4.13: *The services with the generated Coordination Delegates*

5. Experimentation and Empirical Evaluation

The experiments that were designed and performed were to evaluate the platform in terms of service coordination as well as the performance of the implemented use case based on the current platform setting. More specifically, what was tested and observed was the coordination between the services of the use case presented in the previous chapter. In addition, the choreography response time, which is the time required to execute the choreography and present the final results to the end user, was measured for different inputs and different data sets. This was done in order to observe how this specific use case scales in terms of execution time and if there are specific factors that add latency to the execution time of the use case in order to provide feedback to the platform development team regarding e.g., scalability.

5.1 Services Coordination Evaluation

As mentioned, the focus of the first test was the coordination of the implemented services and choreography. The Coordination Delegates are the components responsible for the services coordination. In order to test and observe the interactions between the services, 100 tests were run. For each test, a different origin and destination was submitted from the user web application and the choreography was being executed. It should be mentioned that the origin and the destination were places in the Swedish region, as the Trafikverket API provides traffic situations and information only for the Swedish road infrastructure. It is acknowledged that the purpose of tests performed within the thesis is not to serve as a proof of the correctness of the platform but rather as a first step to showcase its capability with different inputs. With this in mind, those 100 tests together with tests presented in the next section give good knowledge on the questions under investigation. At this phase, having more tests was not important as they would not add any extra knowledge to the knowledge already obtained from these 100 tests. They certainly would not help to prove the correctness of the platform and they were not needed to showcase its capabilities. Rigorous test plans are under development with considerations on more platform components and more complex use cases.

During these tests, the sequence of the method invocations and as a result the interactions between the services were logged in an external file, the same for each test. When a method was invoked, the name of the method was logged in the file right below the previous method that was invoked. A script, able to scan the log file, was created in order to check if the methods were executed with the correct order and the services coordination and interaction was done according to the flow presented in the choreography diagram. Figure 5.1 illustrates the

coordination logic and invocation flow based for the use case considered. It should be noticed that this is for the purpose of explanation and there is no such flow chart to be used by the platform. Instead, the information on coordination logic and the invocation flow are specified in the choreography diagram. The synthesis processor takes in the choreography diagram and generates Coordination Delegates that are responsible to make sure the invocation flow is followed exactly. This literarily means that no matter what the origin and destination is, the method invocation should follow the exact flow. Since the choreography designer knows in detail the services and their methods and knows from the choreography diagram what interactions and method invocations should be expected and needed, it is not hard to produce this flowchart himself.

For all 100 cases, the sequence of the method invocations was matching the pattern or flow of figure 5.1. What is more, in every case the choreography was completed successfully and a route was presented on the map of the web application along with the traffic situations for this route. This fact showed that the services coordination was flawless and that no undesired interactions occurred between services, showing that the generated CDs were indeed able to control and secure the services interactions correctly and complete the goal of the choreography for the needs of this use case. As mentioned before, this was a first step to demonstrate the capability of the platform, and future works on implementing a more complex use case involving a larger number of services are under consideration to further demonstrate the capability of the platform and for quantitative and qualitative testing results.

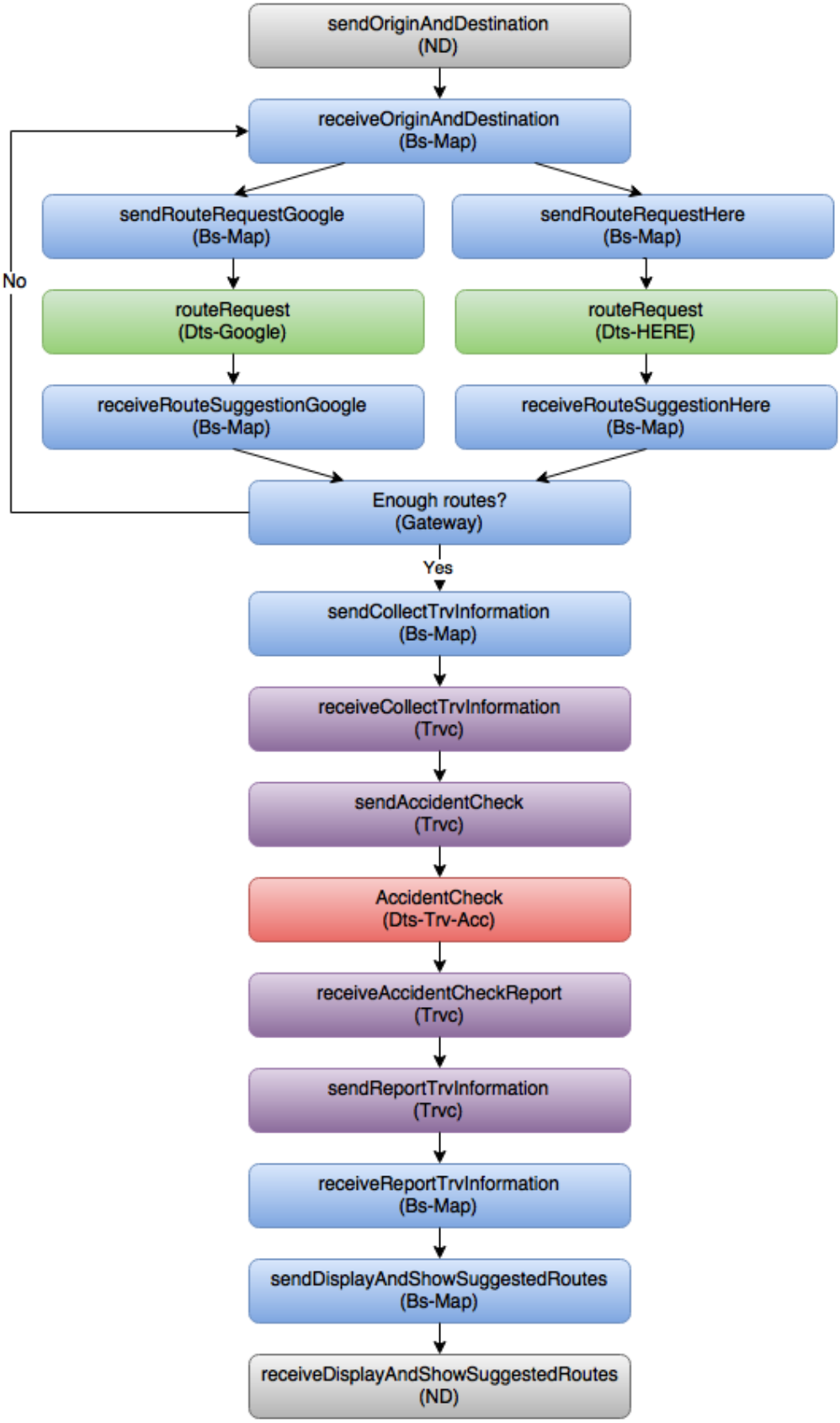


Figure 5.1: *Flow of the invocation of the services methods*

5.2 Response Time Evaluation

Since the coordination between the services was successful and each choreography execution was completed by providing a route to the user, it was essential to check how the implementation of the use case scales for different origins and destinations and for small or larger datasets. In order to do this, the different factors that affect the response time were identified. Once this was done, the time that was required to complete the choreography and present the output to the end user was measured under different conditions.

The response time depends mainly on three factors. The first one is the time that takes to invoke the required methods, coordinate the implemented services and pass the required data between them. These are all actions that are performed by the CDs. Notice that for the time being, the only component included were CDs while the introduction of Adapters, BCs and SFs will surely introduce more latency and will need to be considered once they are available.

The second factor is the time that takes to send a request to the external APIs and receive a response from them. More specifically, for each choreography execution the Google and HERE APIs are queried once, while the requests sent to the Trafikverket API depend on the number of alternative routes that have to be examined. In any case, as set at that time, the maximum number of alternative routes was kept four and the minimum was two (one route from each API) and thus, the requests to the Trafikverket API could not be lower than two or greater than four. It has to be stressed that the effect of these two factors on the overall latency cannot be controlled by the services or the choreography developer. This means that in this case and for these factors the developer cannot intervene to achieve lower latency values.

The third and last factor is the amount of data that has to be processed and how efficiently it is processed. This data includes waypoints and traffic situations. For each route received from the two routing APIs, the legs and maneuvers, mentioned in sections 4.1.2 and 4.1.3, have to be converted into waypoints to comply with the defined data structure. The same procedure applies for the traffic situations received from the Trafikverket API. They need to be processed and converted according to the data structure used. Generally, having longer routes (in km) does not necessarily mean that they contain more waypoints or traffic situations. This is the only factor that the services developer can affect and influence and depends on how efficient is the way that the returned data from the external APIs is processed.

The first step was to measure the response time by running random tests between different locations. For this reason, a list was compiled including the 130

largest urban areas (städer) in Sweden, according to their population. From this list an origin and a destination was being chosen randomly and 500 tests were executed. This number was chosen due to restrictions regarding the number of times that the APIs can be queried during a day or a month. Since the routing APIs are developed for commercial purposes and a free trial key to query them was used, unlimited access was not possible. However, it should be mentioned that the number of the tests was adequate enough and resulted in random routes covering the whole Swedish region. There were large routes running from north parts of the country to south ones and smaller routes inside urban areas, for example from Solna to the center of the Stockholm city or Nacka.

For each test, the total number of waypoints for all the alternative routes, the traffic situations for all the alternative routes and the response time of the choreography were measured. For the measurement of the response time or execution time, a timer was set starting counting from the time the user submits the data from the web application until the final routing information is presented to the user. The total number of waypoints and traffic situations were kept on simple counters and this information along with the response time was stored in an external file. All the data from these tests was stored in the same file which could be accessed to obtain the results and visualize them into meaningful graphs.

The measurement unit for the execution time was chosen to be milliseconds with the purpose of future extension. In the current stage, it should be kept in mind that this is just a first simple use case used to verify the platform while future applications will introduce highly responsive functionalities where latency requirements will be from high to very high, even to milliseconds level. For example, in the case that traffic guidance at intersections is needed, latency will be extremely important. Furthermore, if the platform is proved to be capable, safety-critical scenarios such as collision avoidance can be considered, a case where milliseconds latency is a must. Therefore, study over the latency into high precision was necessary even from the very beginning of this work though it was not absolutely important in the current use case. What is more, using seconds as the measurement unit would be too coarse and would lead to results difficult to be interpreted, since small differences in the latency would not be identified. On the other side, using microseconds would be too fine and would add extra complexity and precision to the final results as well as unnecessary confusion when trying to extract valuable conclusions, which was not needed at least at that point.

One last thing that needs to be cleared before moving on with the tests is that as stated previously the latency requirements for this use case are not high, since the performed tasks are not time-critical. However, from a user experience perspective, having high execution times may disappoint the end users and turn them away from the application. While it is out of scope for this thesis, the thesis

provides statistical results for application designers and developers and it is up to them to determine what the users believe to be an acceptable response time.

Moving on with the tests, what was interesting to get from them was not only the variation of the response time but also the variation of the number of waypoints and traffic situations. This would give an indication of the minimum and maximum number of waypoints and traffic situations that normally need to be processed by the services when searching for routes in the Swedish region, as well as their distribution. The results of these tests are presented in the next three figures, each one containing a boxplot and 500 points visualizing the distribution of the data. In essence, each point is a test result.

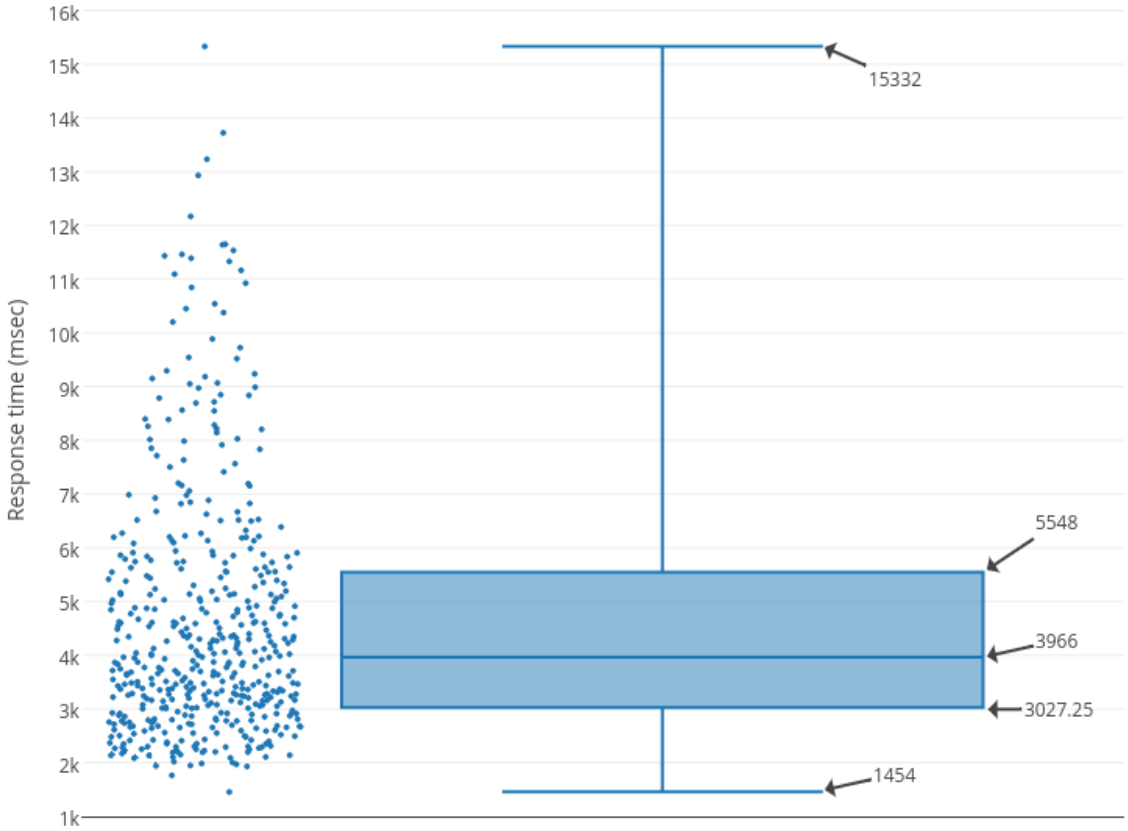


Figure 5.2: Choreography response time distribution for 500 random tests measured in msec

Figure 5.2 shows the distribution of the response times for the 500 random tests. The boxplot organizes the data into four equally sized groups. Each group contains 25% of the total number of the response times, which is 125 and gives an indication of the distribution of the response times. As shown, the median value is 3966 msec, which means that 50% of the response times was above this time and the other 50% below it. More specifically, 25% of the response times is between

1454 msec and 3027 msec (quartile group 1), another 25% is between 3027 msec and 3966 msec (quartile group 2), another 25% between 3966 msec and 5548 msec (quartile group 3) and finally the last 25% of the response times is between 5548 msec and 15332 msec (quartile group 4). As it is observed, the response time can get extremely high in some cases compared to the others.

The first step to understand why and when these high response times are occurring was to check the distribution of the number of waypoints and traffic situations that are processed by the services in each choreography execution during these random tests. It was suspected that for a larger number of waypoints and traffic situations the response time should get higher. What is more, by checking the number of waypoints and traffic situations one could see the range of these values and understand from their distribution what is the number of waypoints and traffic situations that usually have to be processed when this choreography is used for getting routes in the Swedish region.

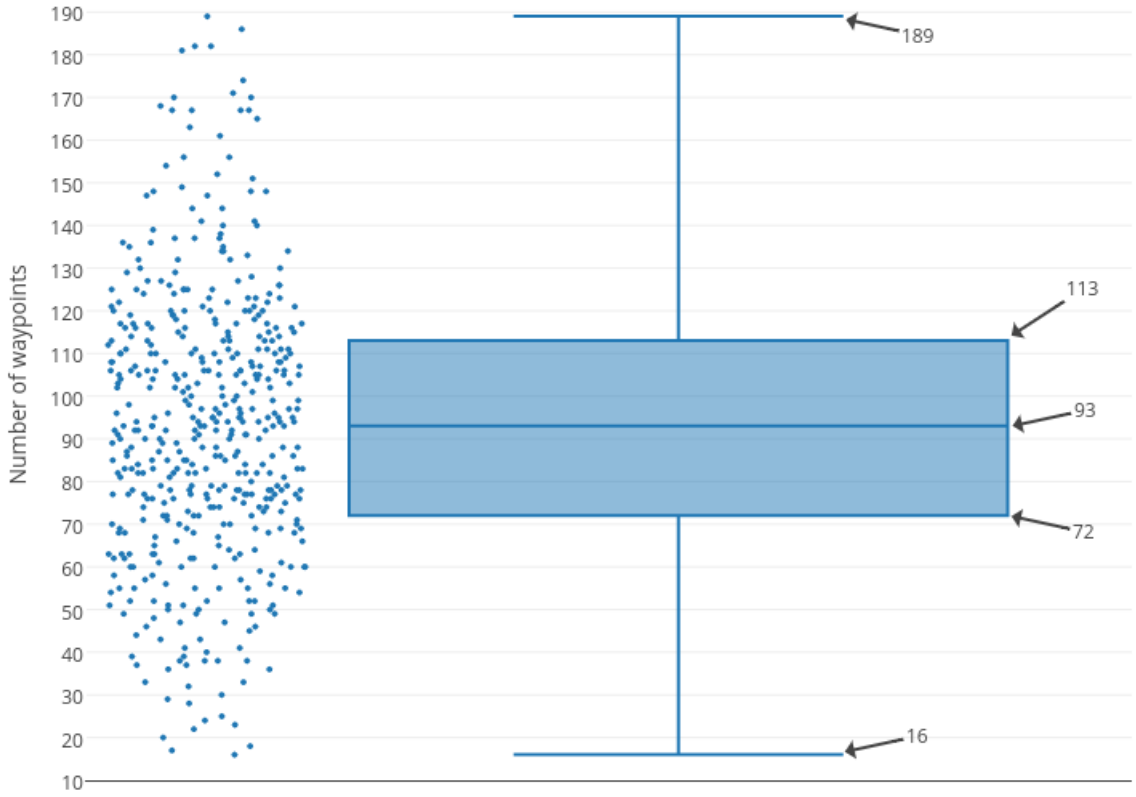


Figure 5.3: *Distribution of the total number of waypoints processed at each test for 500 random tests*

Figure 5.3 shows the distribution of the total number of waypoints that had to be processed for each execution. The total number of waypoints is the sum of the waypoints of each alternative route during a choreography execution. For

example, if during a test four routes are found with each route containing 5 waypoints, then the total number of waypoints is 20.

As depicted on the above boxplot, in half of the tests the number of waypoints that had to be processed was between 16 and 93 and for the other half this number is between 93 and 189. In 50% of the tests, the number of waypoints that had to be processed was between 72 and 113.

The next boxplot, in Figure 5.4, shows the distribution of the total number of traffic situations for all alternative routes that occur during a choreography execution. As shown, the median value is 549 traffic situations. The range of the first quartile is 0-209.25, for the second one is 209.25-549, for the third one is 549-995.75 and for the fourth quartile the range is between 995.75 and 3138. It can be observed that as we move from lower to higher number of traffic situations the density of the points becomes lower.

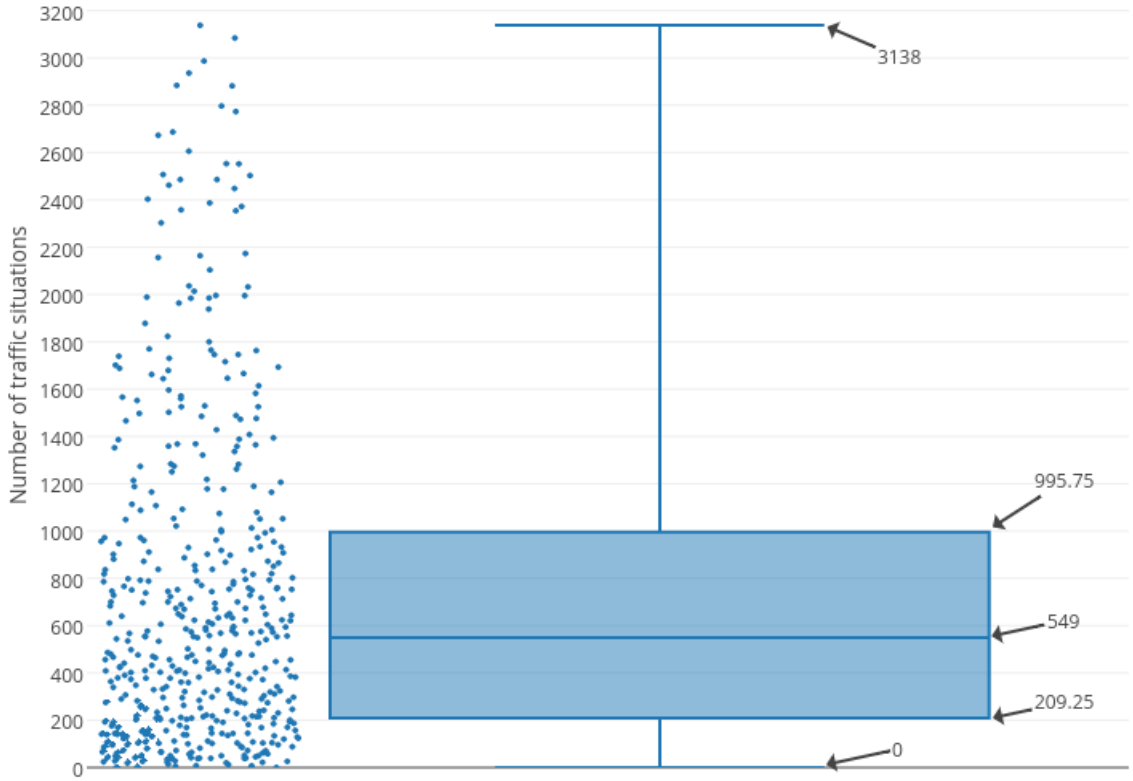


Figure 5.4: *Distribution of the total number of traffic situations processed at each test for 500 random tests*

If a closer look is taken on the exact results that are presented in Appendix A.2, it is shown that there is a great variation in the response times and very large response times may result because of the combination of a large amount of waypoints and a large amount of traffic situations. For example, when having 167

waypoints and 2387 traffic situations the response time was 11650 msec, while on the other side when having to process 24 waypoints and 0 traffic situations the response time was 2019 msec. Thus, what had to be examined was how each of these two values affect the response time and contribute to its variations.

For this reason, three more series of tests were conducted. The first series of tests were to measure the response time of the choreography when the data that has to be processed is the minimum possible. This means that during these tests the origin and destination submitted from the user web application were predefined and were not altered during every execution in order to keep the number of the total waypoints to four, which is the lowest possible, and the number of the traffic situations to zero. As a result, it was possible to get a valuable insight regarding the time required to execute the choreography and how it varies when almost no data has to be processed. In other words, the third factor (process of waypoints and traffic situations) which affects the response time was eliminated and the results depended almost only on the other two, which are the time that takes to invoke the required methods, coordinate the implemented services and pass the required data between them and the time required to query the external APIs. It should be reminded again that the services developer cannot intervene and reduce the latency caused by these two factors. 400 tests were executed and the response time of the choreography as well as the response time of the external APIs for each one of them was calculated and is presented in the following three figures. Again, each point visualizes a single test result.

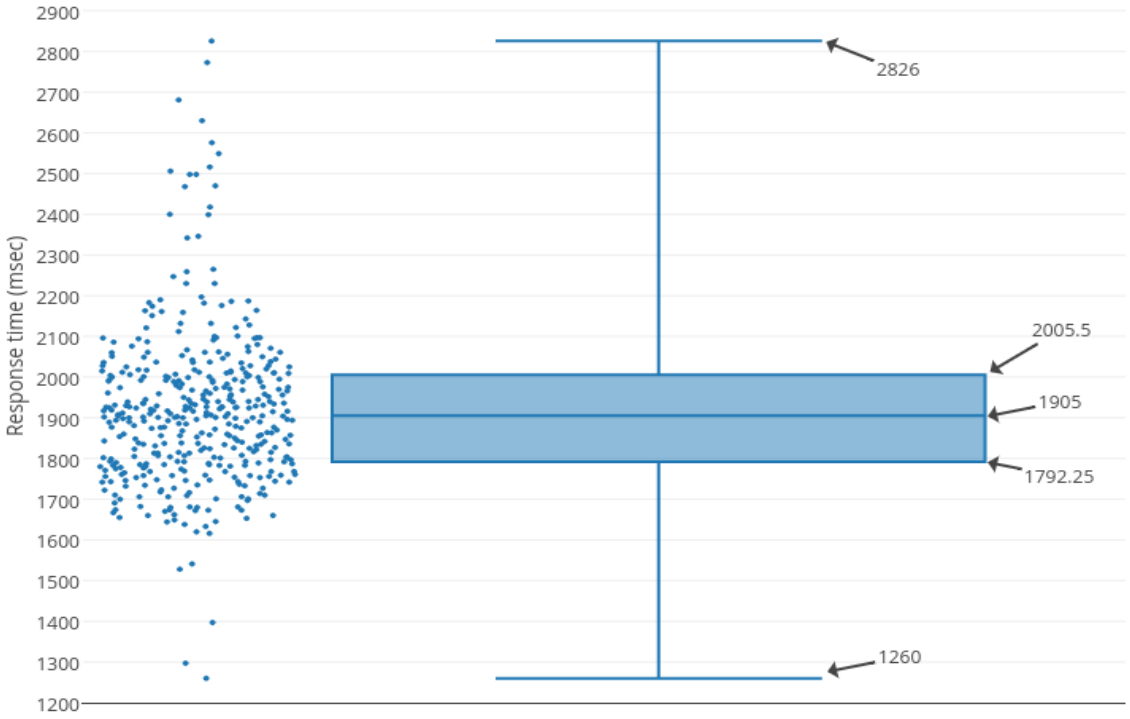


Figure 5.5: *Choreography response time distribution for 400 tests with 4 waypoints and 0 traffic situations*

As shown in Figure 5.5, 50% of the response times for the whole choreography execution was concentrated between 1792 msec and 2005 msec, while by observing the points distribution it could be derived that the response time for most of the tests was between 1500 and 2100 msec. The mean value of the response time was 1919 msec.

Figure 5.6 visualizes the sum of response times for the Google and HERE APIs for the same 400 tests. As it is shown, the time that is required to query these two APIs and receive a response is only a small part of the time required to run the whole choreography with the minimum possible data. The mean time is 253 msec.

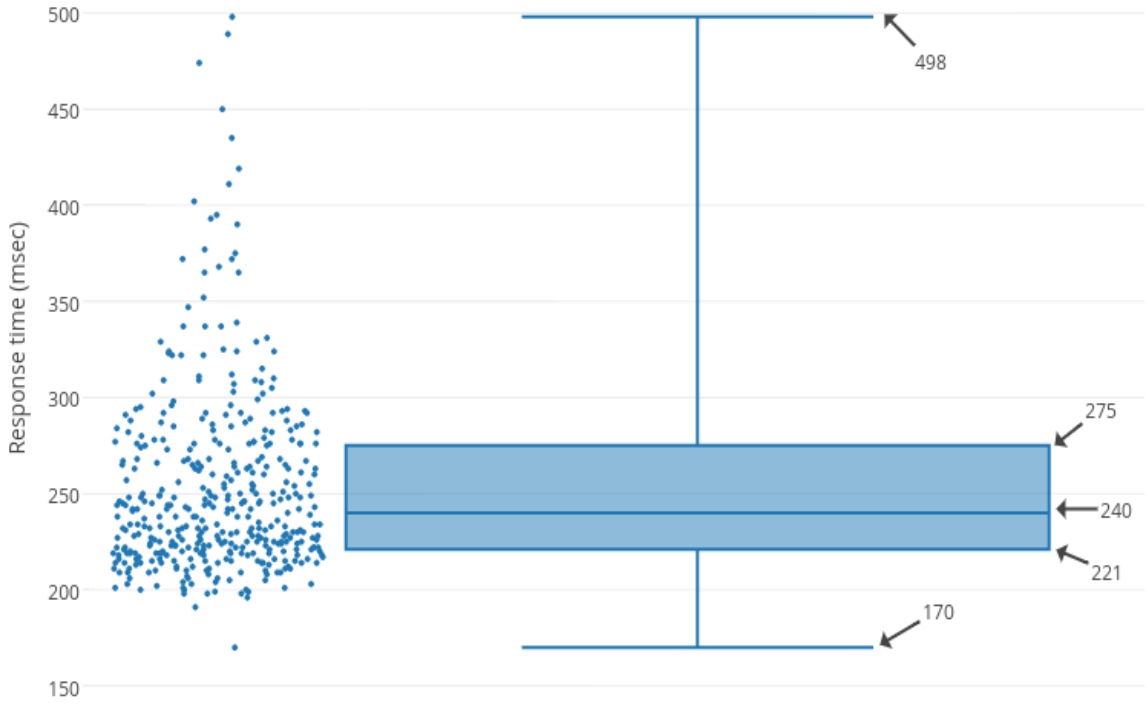


Figure 5.6: Google and HERE APIs response time distribution for 400 tests with 4 waypoints and 0 traffic situations

Figure 5.7 shows the distribution of the Trafikverket API response time for the same 400 tests. It is shown, that even in the worst case the time that is required to get an empty response from this API when there are no traffic situations is totally insignificant compared to the overall execution time of the choreography. The mean Trafikverket response time for these 400 test is 47 msec. It has to be mentioned that the numbers presented in this figure are the result of querying the Trafikverket API twice, one time for the route received from the Google API and one time for the route received from the HERE API.

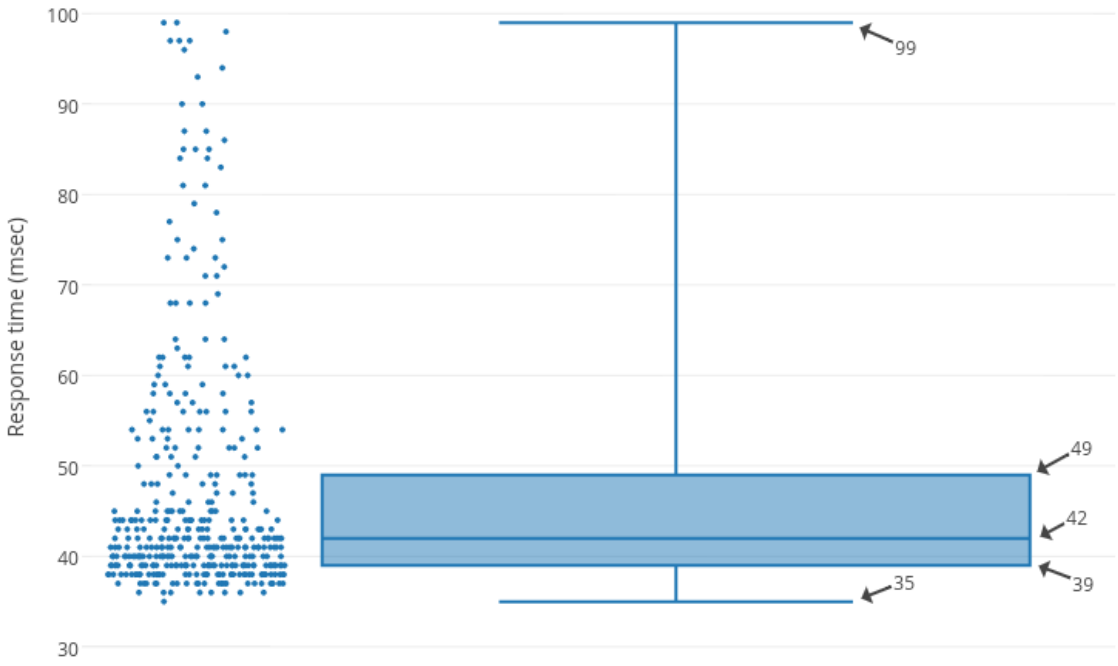


Figure 5.7: Trafikverket API response time distribution for 400 tests with 4 waypoints and 0 traffic situations

If the findings from the above three figures are combined together, it can be calculated that the mean time that takes to invoke the required methods, coordinate the implemented services and pass the required data between them is around 1619 msec. By comparing these values to the previous ones obtained by the random tests in figure 5.2, it was easy to conclude that the latency inserted by the CDs was not what greatly increased the execution time of the choreography and certainly not the reason of the high variation in the execution times. This can be considered beneficial as the performance of the CDs is not something that can be improved from the services developer point of view. It also shows that the number of the waypoints and the number of traffic situations that have to be processed, heavily affects the variation of the choreography's execution time. However, it is still not clear if this is because of the way the data is processed on the local services or because the APIs need more time to respond back when they have to send larger datasets.

The aim of the next series of tests that were conducted was to show more accurately how the number of waypoints affects the response time and if this is due to the way the local services process the waypoints or due to the API calls. For this reason, 500 choreography executions were completed by choosing the origin and the destination randomly from the largest 130 Swedish urban areas. However, the request to the Trafikverket API was modified in order to return always zero traffic situations. This was done to make sure that the response time will not

include the time taken to process any traffic situation. During the 500 executions, the number of waypoints and the response time of the choreography as well as the sum of the response time of all the external APIs were recorded. As it was expected, some tests resulted in the exact same number of waypoints but slightly different response times for the choreography and the API calls. For these results, the mean response time was calculated and used. The results are presented in the below figure where each blue point represents a set of numbers of waypoints and a choreography response time while each orange point represents the APIs response time.

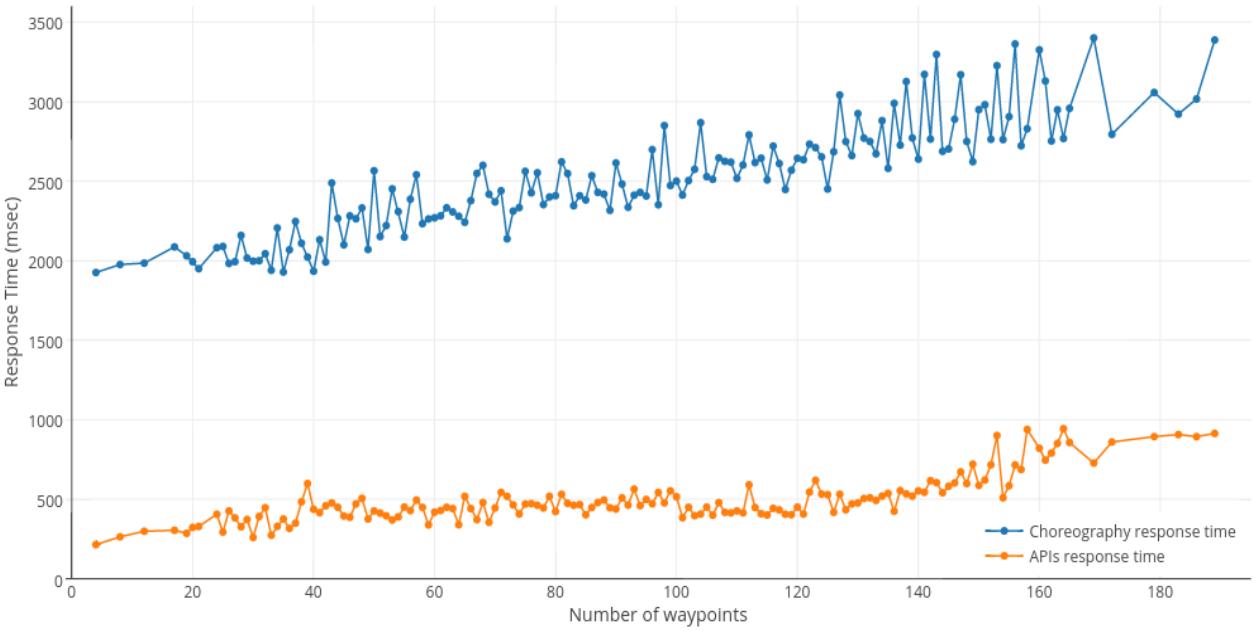


Figure 5.8: *Response time for different number of waypoints when there are no traffic situations*

As it was expected, indeed the number of waypoints affects the response time of the execution of the choreography and the way it varies, since the response time was gradually increased for larger amounts of waypoints that need to be processed. However, even in the worst case shown in the figure, where we have the largest amounts of waypoints, the response time hardly climbed above 3400 msec. Thus, combining the previous tests, where only four waypoints and no traffic situations had to be processed, with these tests, where only waypoints had to be processed, we can see that the latency introduced from the waypoints is quite noticeable. To explain this more, the mean response time when almost no data had to be processed was 1919 msec (the actual range is between 1260 and 2826 msec) and during these tests the max response time was around 3400 msec. This implies that even in this worst case the mean latency added (from both the local

services and the API calls) was around 1500 msec and to be more specific it was for sure lower than 2140 msec and greater 570 msec.

What is more, it can be seen that the APIs response time is gradually slightly increased but not at the same rate as the whole choreography response time. This leads to the observation that the largest part of the increase in the choreography execution time is due to the time that is required to process the waypoints returned by the Google and HERE APIs. For a larger number of waypoints, the two routing APIs need more time to send back a response, because of the larger number of waypoints that needs to be returned, but this is not enough to significantly increase the latency of the overall choreography execution. It should be mentioned that the APIs response time presented in the previous figure is the sum of one call to the Google API, one call to the HERE API and two calls to the Trafikverket API. However, this slight increase in the orange line is because of the slight increase of the response time of the Google and HERE APIs, since the Trafikverket API was always replying with the same empty response.

The last series of tests that were conducted were to examine how the number of traffic situations affects the choreography response time. During these tests, the choreography was executed again 500 times. This time, the request to the Google and HERE APIs was always for the same origin and destination because we wanted to keep the number of waypoints the minimum possible which is four. However, the query to the Trafikverket API was requesting traffic situations for different routes in order to reply every time with different number of traffic situations. The number of situations that were returned and had to be processed were in the same range as the one presented in figure 5.4. As it was expected, some tests resulted in the exact same number of traffic situations but slightly different response times. For these results, the mean response time was calculated and used. Figure 5.9 contains the results of these tests, the response time for the whole choreography as well as the sum of the APIs response time.

As expected, the figure shows that the number of traffic situations that have to be processed is crucial to the overall execution time of the choreography, and is the parameter that affects the response time more than any other factor. However, we have to keep in mind that this number can be much higher than the number of waypoints and reach values like 3000 or more. As observed, for these values the choreography response time may be around 10000msec. Consequently, it was safe enough to conclude that the latency added from a high number of traffic situations is greatly decreasing the overall performance of the response time and can add latency that is surely highly noticeable to the end user. For example, during these tests the worst case was a response time of almost 11000 msec. This means that given the fact that the mean response time when almost no data had to be processed was 1919 msec (the actual range of the tests

conducted was between 1260 and 2826 msec), in this specific case the latency inserted from the process of waypoints (from both the local services and the API calls) was surely between 8100 and 9700 msec. However, it should be pointed out that as shown in figure 5.4, in 50% of the random tests executed, the number of traffic situations was below 600. For such a number the execution time rarely exceeded the 4000 msec, when almost no waypoints had to be processed.

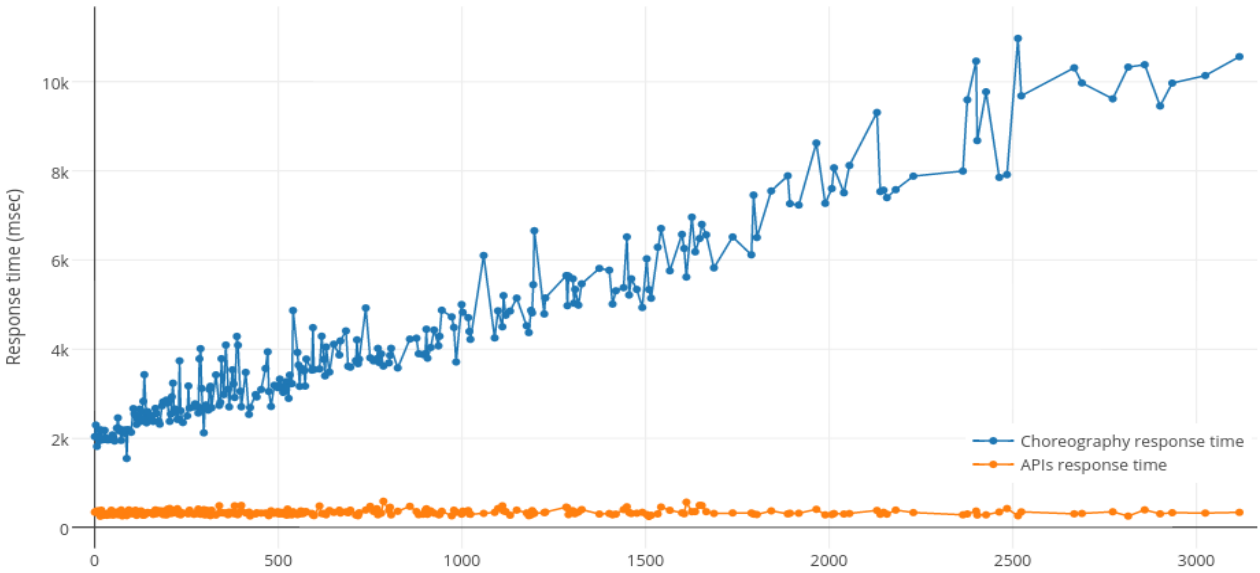


Figure 5.9: *Response time for different number of traffic situations when there are almost no waypoints*

What is worth mentioning, is that the APIs response time does not follow the increase of the overall choreography response time. In fact, the number of traffic situations that needs to be returned from the Trafikverket API does not seem to affect the time that this API needs to prepare and send back a response. Therefore, it can be stated that the latency added to the overall response time of the choreography is almost due to the time required to process the traffic situations returned. In Figure 5.9 the APIs response time is the sum of the time that is needed to query once the Google and the HERE APIs and the time needed to query the Trafikverket API. The number of times that the Trafikverket API had to be queried was in the range from two to four, as it was depended on the number of routes returned by the two routing APIs. Surprisingly, a single query to the Trafikverket API almost never exceeded 30 msec.

As noticed from all the tests and choreography executions, under the current use case, the coordination of the services itself and the query of the external APIs is not time consuming when compared to the overall execution time of the choreography. What increases the response time and greatly contributes to

its high variation is the processing of the data that is retrieved from the APIs and their conversion to fit into the data structure that is defined. As a result, it can be claimed that the choreography execution itself and the CDs as well as the APIs requests do not impose latency that can greatly affect the final result, when compared to the latency added by the third factor identified. In addition, when the number of the traffic situations is extremely large and is combined with a high number of waypoints, the response time is very negatively affected. The processing of the waypoints and the traffic situations is part of the business logic and the implementation of the services methods. For better results with lower response times what should be improved is this internal business logic and the way the data is processed, something that the services developer is responsible to do.

Other than that, considering the number of the web services and the number of the methods that have to be invoked, in most of the tests conducted the response time was fairly good when keeping in mind that the task is not time-critical. We should not forget that during the random tests conducted and as shown in Figure 5.2, in 75% of the tests the response time was below 5548 msec. However, this does not change the fact that extreme values and response times may occur. Usually, this is because of the combination of a large number of waypoints and a large number of traffic situations.

One last thing that should be mentioned is that the API calls were done using specific network routes free of congestion. Using different network routes to reach these APIs will surely lead to different time results. As an indication, for the specific API response time tests conducted and presented in figures 5.6-9, the roundtrip time to send an ICMP echo-request to the Google API and receive back an echo-reply was on average 5msec. The same time for the Trafikverket API was on average 15 msec. Calculating the same for the HERE API was not possible since ICMP traffic was dropped before reaching the destination.

6. Conclusions

The thesis presented the web services choreography architecture to design web services and introduced the CHOReVOLUTION platform, a platform that aims to automatically synthesize dynamic choreographies. A simple use case inside the ITS domain, the first ever created and deployed over the platform, was designed in order to prove the functionality of the platform at least for this specific use case and show how it is able to successfully take care of the interactions between the services developed. The evaluation of the use case was divided into two parts. The first part was to monitor and verify the interactions between the services that were designed and deployed for this specific use case. Responsible for these interactions are the Coordination Delegates, which are components generated by the platform. The second part was to evaluate the use case itself in terms of the choreography execution time in order to identify the factors that contribute to the overall execution time and explore why increased latency values may occur.

6.1 Discussion

By using the results collected in Chapter 5, the scientific questions of section 1.3 could be safely answered as per below.

Q1: Is this specific platform able to automatically produce the required components for the designed use case and choreography?

As shown during the deployment of the use case over the platform, the platform was able to produce the required components for the designed choreography. During the deployment, no warnings or exceptions occurred and the whole process was completed successfully. In the end of the process, the CDs were automatically generated and interposed between the services wherever needed and the choreography was ready to be executed. It has to be pointed out that the successful automatic generation of the components under this use case does not mean that the generation of the components will surely be successful for other use cases or even for the same use case under a different setup than the one described in chapter 4.

Q2: Are the generated components able to accurately coordinate the interactions between the services for the specific use case according to the specific

coordination logic; or interactions, which are not desirable, occur leading to unpredictable behavior of the use case? It is assumed that:

i) Any external interference is absent and does not occur during the choreography execution.
ii) The deployment environment and the services are stable and cannot be altered while the choreography is running

It was shown that the generated components are capable of performing the coordination between the services as specified for the use case and its setup under consideration in this thesis. During all the tests and executions of the choreography, the sequence of the services methods that were executed was the desirable one (presented in Figure 5.1) and the correct final result was delivered to the end user through the web application. The behavior of the use case was as expected and defined by the choreography diagram and no undesired interactions occurred in the system. It has to be noticed that any outside intervention or interference was excluded and the same software and hardware setup was used during the tests. Moreover, it should be acknowledged that during the thesis, the services coordination was found to be successful only under the current use case and for the current platform setup. The answer to this question applies only to the present use case and cannot be generalized in order to draw conclusions generally for the platform or for other future use cases. The current use case serves as a first step to showcase the capability of the platform and not prove the correctness of it.

Once more, it should be made clear that the flowchart in Figure 5.1, that led to the answer of this question, was not produced mechanically from the choreography diagram. Since it was produced by the designer of the choreography, it can be concluded that this kind of flowchart should be used as a validation tool as long as there is an expert who can produce it, and the complexity of the choreography and the generated call sequence does not exceed the competence of the expert. Thus, thought it might serve as an efficient validation tool for simple choreographies, like the one presented in this thesis, for larger choreographies where the complexity of the generated call sequence can be higher, manually producing such a flowchart could be a difficult and prone to errors procedure. It would be beneficial, if the synthesis processor could automatically produce a flowchart for subsequent validation.

Q3: What are the different factors that may affect the latency of the designed choreography?

As it was identified from the study of the platform, the design of the use case and during the experiments, there are three factors that affect the overall execution time. The first one is the use of the coordination delegates. These are responsible for invoking the required service methods, control the services coordination and pass the required data between them. The second factor is the requests to the external APIs as described in chapter 5. For each choreography execution three different APIs have to be queried. These queries contribute to the overall latency of the choreography. Finally, the last factor is the amount of data that has to be processed and how efficiently it is processed. This data can be divided into waypoints and traffic situations. As it is natural, the larger the amount of data to be processed, the higher the execution time of the choreography is.

Q4: Are there any of these factors that can be controlled by the developer of the use case in order to decrease the latency inserted by them and improve the overall latency of the system and the choreography?

As it is easily understood from the answer of question 4, the only factor that can be controlled by the developer of the use case in order to decrease the latency inserted by this factor and improve the overall latency the choreography, is how the data is processed. This is possible, since the processing of the data is part of the services' business logic, something implemented by the use case developer. Having decreased latency may be possible by rewriting the code that processes the data that is returned from the external APIs in a more efficient way. Unfortunately, the latency introduced by the other two factors cannot be decreased from the services developer perspective since none of them is an issue of the implementation of the current use case presented.

Q5: What is the overall latency (or execution time) of the system for the use case that will be demonstrated and how much does it vary for different choreography executions?

As shown from the random tests conducted in chapter 5 and presented in figure 5.2 and Appendix A2, the latency of the system depends on the input from the user app. The detailed latency values can be viewed under figure 5.2 or in Appendix A2. It should be mentioned again than in 50% of the random tests executed, the latency was between 3027 and 5548msec. However, sometimes the execution time could be really high and in extreme cases it could go beyond 10000msec. The lowest latency value observed during these random tests was 1454msec and the highest one 15332msec. This shows that the overall execution

time can largely vary. It has to be pointed out, that the latency results obtained here cannot be used to calculate or predict the latency of other use cases or even the latency of a slightly altered version of the use case presented. New tests will have to be conducted.

Q6: Which are the main contributing factors in the variability of the choreography's execution time?

As noticed from the experiments, the execution time of the choreography can greatly vary and sometimes can be really high. The reason that greatly limits the current use case, is mainly responsible for the high variability of the execution time and sometimes results in very high execution times is the amount of data that has to be processed by the local services.

As shown, the data can be divided into two categories depending on its type. The first type is the waypoints and the second is the traffic situations. When no waypoints and traffic situations have to be processed, the choreography execution time, as shown in figure 5.5, is between 1260 and 2826msec. What is more, in 94,5% of the tests conducted and presented in the same figure, the execution time is between 1600 and 2300msec. By comparing these results to the ones from figure 5.2, it can be stated that in the absence of waypoints and traffic situations the execution time does not greatly vary. As a consequence, what mainly contributes to the variation of the execution time is the number of waypoints and traffic situations that have to be processed.

These two factors were separately tested and results from these tests were presented in figures 5.8 and 5.9. This was done in order to confirm that indeed these are the mainly contributing factors in the variability of the choreography's execution time. By observing the results from these tests, it could be concluded that the execution time of the choreography is greatly affected by these two factors. The more traffic situations or waypoints, the higher the execution time is. Hence, this leads to a great variation of the execution time, since the amount of waypoints and traffic situations can also greatly vary as shown in figures 5.3 and 5.4, depending on the input submitted to the user application.

One last observation, that has to be mentioned, is that higher choreography execution times occur when the amount of waypoints or the amount of traffic situations that has to be processed is also higher, as shown in figures 5.8 and 5.9. It could be safely stated that a combination of a high number of waypoints and high number of traffic situations will lead to a high execution time which is mainly because of the amount of data that has to be processed and not because of the

API calls. Fortunately, as mentioned previously, the latency inserted by the amount of data that has to be processed can possibly be decreased by rewriting the code that processes the data that is provided by the external APIs in a more efficient way.

Since the questions from section 1.3 are answered, one more fact that needs to be mentioned again is that the platform at the time of the use case implementation was still under development. During this thesis and for the examined use case it was shown that the platform can complete its primary mission which is the coordination of the web services. However, there are more functionalities that are currently developed which will make it a platform that will be able to leverage concepts like the Future Internet, the Smart Cities and the Internet of Things. For example, the Binding Components, mentioned in section 3.6, will make the interactions of different types of web services possible. As a result, data heterogeneity will not be a problem anymore. This is really important when moving towards the Internet of Things since the more lightweight protocols used, like the Message Queuing Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP), are completely different from traditional ones, like SOAP and REST. It has to be mentioned that these new components are likely to increase the latency even more. However, no safe assumptions can be made without testing a new use case where these components will be present.

What was developed in the context of this thesis project and the discussion above essentially showcases the benefits of the platform and the choreography architecture into the ITS domain. Different services and information from different domains and sources are able to be combined together and collaborate to manage better the traffic and provide accurate real time data and decisions to drivers. In this specific use case, three different data sources were combined in order to provide the user with essential routing information for the Swedish region. By presenting all the different components of the platform, even the ones that are still under development, it is fairly easy to get an insight of how this platform can serve the ITS domain. Since we are moving towards the Internet of Things more traffic elements, like traffic lights or bridges, can be leveraged and get interconnected in order to form web services and applications which will be able, by collaborating and interacting together, to provide novel solutions to already existing problems like the driving safety and the traffic management. Since the platform, at the time the thesis was conducted, was still under development and the implemented use case is a simple one, more work is needed to safely and accurately answer if this platform and the choreography approach can clearly benefit the ITS domain. A more precise answer to this question will be given with the development of the platform to include more components and the extension of the work performed in this thesis to include more complex use cases. As a general conclusion from the work performed so far, the choreography-based solution and the platform which

was used, were demonstrated beneficial to application development in the ITS domain.

What has not been pointed out but is clearly obvious from the flow of the use case implementation, is that, in essence, this platform is a tool that greatly simplifies the work of the web services developers. All that is needed is to create the web services almost the way traditional web services are developed and describe their interactions through the choreography diagram. The design of the diagram is a fairly easy process once the interactions are defined. In this way, the developers are able to focus on the web services, their functionalities and the business logic rather than dealing with complex languages, such as the BPEL, that are used for the orchestration of web services. The implementation of the interaction and the communication between the services, which otherwise would be a time-consuming task, is totally taken care by the platform without requiring any action from the developer. In fact, there is no need for the developers to even understand thoroughly the way the platform works and how the related components are generated.

6.2 Limitations

The biggest limitation faced regarding the implementation of this project was the development of the necessary web services, since it was done in parallel with the development of basic and primary features of the platform. To put it in another way, when the implementation of the services started the platform prototype was not ready and the guidelines for the way the services should be implemented were not clear, resulting in a very time consuming procedure. During the implementation and according to the needs of the platform development process, a couple of different design approaches for the web services were implemented before ending up on the final one used. The one used is the simpler one in terms of design and does not introduce a lot of differences when compared to traditional web services designs. This was done to make the development of the necessary web services as easier as possible for the developers.

The second limitation is that no cloud system was available to deploy the services and run tests. Instead, the required servers that host the services and the Coordination Delegates were deployed over a retail laptop, which has less computational power than normally servers have. What can be assumed from this fact, is that the performance of the choreography would be better if deployed over a cloud system. However, this is just an assumption that has to be proved.

It should also be mentioned clearly that for each API that was used, an API key had to be obtained. The APIs offered by Google and Nokia are essentially

commercial APIs. Although that this means that in order to obtain full access and be able to submit unlimited queries a fee has to be paid, they also offer some free trial API keys that were used for this project. This comes with limitations regarding the number of queries that can be submitted per day or per month. Thus, the number of the choreography executions during the tests had to be limited in order not to exceed the number of allowed queries to these APIs. However, the final number of the execution times of each experiment conducted did not limit the research within this thesis as we were able to get routes covering the whole Swedish region, from really short ones to really long ones. The usage of external and commercial service will add cost on the application itself that should be dealt by the future business model.

Finally, what needs to be pointed out and clarified again is that every conclusion extracted from the deployment and execution of this use case cannot be generalized to safely draw conclusions about other potential use cases or the platform in general. Even in case of expanding the current use case and inserting just one service, it cannot be surely stated that the platform will be able to flawlessly produce the required components and handle correctly all the new interactions. As mentioned, the use case in this thesis serves as a first step to showcase the capability of the platform, while in the future rigorous test procedures will be developed before the platform release. In general, it is acknowledged that all the conclusions drawn are valid for the behavior of the platform under this specific use case and the specific platform setup, described in chapter 4, and the measurements and results that are taken from this use case are not expected to be generalized in a trivial way to other use cases. On the other hand, the methodologies applied within this thesis can be used for further investigation of the platform.

6.3 Future Work

Having as a starting point what was implemented and tested in this thesis project, there are various directions one can follow to expand this work. One direction could focus on the platform and the validation of components that are still under development, such as the Binding Components and the Security Filters. This can be done by modifying the services of the current use case or even better expanding it and building new services which will use different protocol types or by introducing security features like the use of credentials to log in before using the web application.

Another direction is towards the ITS domain. One could solely focus on the creation of algorithms for better evaluation of the returned routes based for

example on eco-aware driving or expand the use case and create services to enable new features like speed suggestions to the driver.

References

[1] Organisation for Economic Co-operation and Development, European Conference of Ministers of Transport, and Transport Research Centre, Eds., *Managing urban traffic congestion*. Paris: OECD : ECMT, 2007.

[2] B. S. Kerner, *Introduction to Modern Traffic Flow Theory and Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[3] "Directive 2010/40/EU of the European Parliament and of the Council of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport OJ L207/1." Aug-2010.

[4] "chorevolution.eu: (Main.WebHome)." [Online]. Available: http://www.chorevolution.eu/bin/view/Main/. [Accessed: 15-Feb-2016].

[5] "Google Maps APIs | Google Developers." [Online]. Available: https://developers.google.com/maps/. [Accessed: 01-Mar-2016].

[6] "Öppet API - Trafikverket." [Online]. Available: http://api.trafikinfo.trafikverket.se/. [Accessed: 01-Mar-2016].

[7] "Google APIs Terms of Service | Google Developers." [Online]. Available: https://developers.google.com/terms/. [Accessed: 04-Oct-2016].

[8] "Terms and Conditions - HERE Developer." [Online]. Available: https://developer.here.com/terms-and-conditions. [Accessed: 04-Oct-2016].

[9] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, Jun. 2003.

[10] E. S. Wilschut, "The impact of in-vehicle information systems on simulated driving performance," *Unpubl. Dr. Diss.*, 2009.

[11] L. Figueiredo, I. Jesus, J. T. Machado, J. Ferreira, and J. M. De Carvalho, "Towards the development of intelligent transportation systems," in *Intelligent Transportation Systems*, 2001, vol. 88, pp. 1206–1211.

[12] Z. Xiong, H. Sheng, W. Rong, and D. E. Cooper, "Intelligent transportation systems for smart cities: a progress review," *Sci. China Inf. Sci.*, vol. 55, no. 12, pp. 2908–2914, Dec. 2012.

[13] "Intelligent Transportation Systems - Connected Vehicle Research." [Online]. Available: http://www.its.dot.gov/connected_vehicle/connected_vehicle_research.htm. [Accessed: 23-Mar-2016].

[14] eSafety Support, "Report on the Progress of the 28 eSafety Recommendations." Jul-2007.

[15] S. Ueda, F. Kanazawa, J. Sawa, S. Suzuki, H. Okada, and T. Maeda, "Research for Practical Use of Smartway and Deployment of ITS Spot Services," presented at the 18th ITS World Congress, Orlando Florida, 2011, vol. 6, p. 10.

[16] O. M. J. Carsten and L. Nilsson, "Safety assessment of driver assistance systems," *Eur. J. Transp. Infrastruct. Res.*, vol. 1, no. 3, pp. 225–243, 2001.

[17] P. J. Zwaneveld, B. Van Arem, E. Bastiaensen, H. Soeteman, and B. Ulmer, "Deployment scenarios for Advanced Driver Assistance Systems in Europe,"

in *Proceedings of the 6th World Congress on Intelligent Transport Systems*, Toronto, Canada, 1999.
[18] W. Barfield and T. A. Dingus, Eds., *Human factors in intelligent transportation systems*. Mahwah, N.J: Lawrence Erlbaum Associates, 1998.
[19] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Inf. Sci.*, vol. 280, pp. 218–238, Oct. 2014.
[20] M. G. Nanda, S. Chandra, and V. Sarkar, "Decentralizing execution of composite web services," in *ACM Sigplan Notices*, 2004, vol. 39, pp. 170–187.
[21] L. A. F. Leite, G. Ansaldi Oliva, G. M. Nogueira, M. A. Gerosa, F. Kon, and D. S. Milojicic, "A systematic literature review of service choreography adaptation," *Serv. Oriented Comput. Appl.*, vol. 7, no. 3, pp. 199–216, Sep. 2013.
[22] "Web Services Business Process Execution Language." [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html. [Accessed: 15-Feb-2016].
[23] "IBM - Business Process Automation - WebSphere Process Server." [Online]. Available: http://www-01.ibm.com/software/integration/wps/. [Accessed: 15-Feb-2016].
[24] "Orchestra: Open Source BPEL / BPM Solution - Orchestra : The Open Source BPEL solution." [Online]. Available: http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome. [Accessed: 15-Feb-2016].
[25] A. Barker, C. D. Walton, and D. Robertson, "Choreographing Web Services," *IEEE Trans. Serv. Comput.*, vol. 2, no. 2, pp. 152–166, Apr. 2009.
[26] F. Daniel and B. Pernici, "Insights into web service orchestration and choreography," *Int. J. E-Bus. Res. IJEBR*, vol. 2, no. 1, pp. 58–77, 2006.
[27] "BPMN 2.0." [Online]. Available: http://www.omg.org/spec/BPMN/2.0/. [Accessed: 15-Feb-2016].
[28] "Web Service Choreography Interface (WSCI) 1.0." [Online]. Available: https://www.w3.org/TR/wsci/. [Accessed: 15-Feb-2016].
[29] "Web Services Choreography Description Language Version 1.0." [Online]. Available: https://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/. [Accessed: 15-Feb-2016].
[30] A. Barker, P. Besana, D. Robertson, and J. B. Weissman, "The benefits of service choreography for data-intensive computing," in *Proceedings of the 7th international workshop on Challenges of large applications in distributed environments*, 2009, pp. 1–10.
[31] D. Liu, K. H. Law, and G. Wiederhold, "Analysis of integration models for service composition," in *Proceedings of the 3rd international workshop on Software and performance*, 2002, pp. 158–165.
[32] M. zur Muehlen and J. Su, Eds., *Business Process Management Workshops*, vol. 66. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
[33] M. Tarek, C. Boutrous-Saab, and S. Rampacek, "Verifying Correctness of Web Services Choreography," 2006, pp. 306–318.

[34] L. F. Herrera-Quintero, F. Maciá-Pérez, D. Marcos-Jorquera, and V. Gilart-Iglesias, “SOA-Based Model for Value-Added ITS Services Delivery,” *Sci. World J.*, vol. 2014, pp. 1–19, 2014.
[35] J.-P. Kim, J.-E. Hong, J.-Y. Choi, and Y.-H. Cho, “Dynamic service orchestration for SaaS application in web environment,” 2012, p. 1.
[36] J. Gacnik, O. Hager, and M. Hannibal, “A Service-Oriented System Architecture For The Human Centered Design Of Intelligent Transportation Systems,” in *Proceedings of European Conference on Human Centred Design for Intelligent Transportation Systems*, Lyon, France, 2008.
[37] “VRA in Brief | VRA.” [Online]. Available: http://vra-net.eu/about-the-vra-network/. [Accessed: 03-Sep-2016].
[38] “Knowledge Center 2.0.” [Online]. Available: http://www.itsa.org/knowledgecenter/knowledge-center-20. [Accessed: 03-Sep-2016].
[39] “Knowledge Center | ITS asia-pacific.” [Online]. Available: http://itsasia-pacific.com/knowledge-center/. [Accessed: 03-Sep-2016].
[40] “ERTICO ERTICO - ITS Europe.” [Online]. Available: http://ertico.com/. [Accessed: 03-Sep-2016].
[41] “IEEE Xplore Digital Library.” [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp. [Accessed: 03-Sep-2016].
[42] M. Bravetti, C. Guidi, R. Lucchi, and G. Zavattaro, “Supporting e-commerce systems formalization with choreography languages,” in *Proceedings of the 2005 ACM symposium on Applied computing*, 2005, pp. 831–835.
[43] M. S. Benabdelhafid and M. Boufaida, “The Need for Formal Compatibility Analysis in Web Service Choreography via an E-Commerce Application:,” *Int. J. E-Bus. Res.*, vol. 11, no. 4, pp. 1–16, 34 2015.
[44] E. Ivanova and T. Stoilov, “Information Technologies in E-Government Solution.”
[45] J. Mendling and M. Hafner, “From WS-CDL choreography to BPEL process orchestration,” *J. Enterp. Inf. Manag.*, vol. 21, no. 5, pp. 525–542, Sep. 2008.
[46] R. De Knikker, Y. Guo, J. Li, A. K. Kwan, K. Y. Yip, D. W. Cheung, and K.-H. Cheung, “A web services choreography scenario for interoperating bioinformatics applications,” *BMC Bioinformatics*, vol. 5, no. 1, p. 1, 2004.
[47] P. M. K. Gordon and C. W. Sensen, “Creating Bioinformatics Semantic Web Services from Existing Web Services: A Real-World Application of SAWSDL,” 2008, pp. 608–614.
[48] J. Seo, Y. Kido, S. Seno, Y. Takenaka, and H. Matsuda, “A Method for Efficient Execution of Bioinformatics Workflows,” in *The 20th International Conference on Genome Informatics, Japan*, 2009.
[49] R. Breu, M. Hafner, B. Weber, and A. Novak, “Model Driven Security for Inter-organizational Workflows in e-Government,” in *E-Government: Towards Electronic Democracy: International Conference, TCGOV 2005, Bolzano, Italy, March 2-4, 2005. Proceedings*, M. Böhlen, J. Gamper, W. Polasek, and M. A. Wimmer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 122–133.

[50] “choreos.eu: CHOReOS: Large Scale Choreographies of the Future Internet (Main.WebHome).” [Online]. Available: http://www.choreos.eu/bin/view/Main/#. [Accessed: 21-Feb-2016].
[51] F. Antonelli, M. Calisti, E. Fernandez, R. Giaffreda, J. Gonzalez, E. Kim, T. Lahnalampi, M. Potts, and E. Salvadori, “Map of technology and business challenges for the Future Internet.” FI-LINKS Consortium, 2016.
[52] “Future Internet Socio Economics - FisaWiki.” [Online]. Available: http://fisa.future-internet.eu/index.php/Future_Internet_Socio_Economics. [Accessed: 21-Feb-2016].
[53] CHOReOS Project Team, “Deliverable 5.6 - Final version and assessment of the CHOReOS IDRE.” Oct-2013.
[54] “Eclipse BPMN2 Modeler.” [Online]. Available: https://www.eclipse.org/bpmn2-modeler/. [Accessed: 25-Feb-2016].
[55] “Business Process Model and Notation (BPMN) Specification.” Object Management Group, Jan-2011.
[56] CHOReVOLUTION Project Team, “Deliverable D2.2 - CHOReVOLUTION Synthesis - First outcomes.” Nov-2015.
[57] M. Autili and M. Tivoli, “Distributed Enforcement of Service Choreographies,” *Electron. Proc. Theor. Comput. Sci.*, vol. 175, pp. 18–35, Feb. 2015.
[58] “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).” [Online]. Available: https://www.w3.org/TR/soap12/. [Accessed: 29-Feb-2016].
[59] CHOReVOLUTION Project Team, “Deliverable 3.1 - CHOReVOLUTION Service Bus, Security and Cloud - First outcomes,” Nov. 2015.
[60] “Google Maps Directions API | Google Developers.” [Online]. Available: https://developers.google.com/maps/documentation/directions/. [Accessed: 21-Jun-2016].
[61] “Overview - Routing API - HERE Developer.” [Online]. Available: https://developer.here.com/rest-apis/documentation/routing. [Accessed: 21-Jun-2016].
[62] “MapQuest Developer Network | Directions API.” [Online]. Available: https://developer.mapquest.com/products/directions. [Accessed: 21-Jun-2016].
[63] “Mapbox API Documentation.” [Online]. Available: https://www.mapbox.com/api-documentation/#retrieve-a-standalone-marker. [Accessed: 21-Jun-2016].
[64] “API.” [Online]. Available: https://api.trafikinfo.trafikverket.se/API. [Accessed: 21-Jun-2016].
[65] “Getting Started | Google Maps Geocoding API | Google Developers.” [Online]. Available: https://developers.google.com/maps/documentation/geocoding/start. [Accessed: 28-Jun-2016].
[66] “Leaflet - a JavaScript library for interactive maps.” [Online]. Available: http://leafletjs.com/. [Accessed: 28-Jun-2016].
[67] “GeoJSON Specification.” [Online]. Available: http://geojson.org/geojson-spec.html. [Accessed: 28-Jun-2016].

Appendix A

A.1 List of the urban areas in Sweden used for the random tests in section 5.2

Alingsås
Arboga
Arvika
Askersund
Avesta
Boden
Bollnäs
Borgholm
Borlänge
Borås
Djursholm
Eksjö
Enköping
Eskilstuna
Eslöv
Fagersta
Falkenberg
Falköping
Falsterbo
Falun
Filipstad
Flen
Gothenburg
Granna
Gävle
Hagfors
Halmstad
Haparanda
Hedemora
Helsingborg
Hjo
Hudiksvall
Huskvarna
Härnösand
Hässleholm
Höganäs
Jönköping
Kalmar
Karlshamn
Karlskoga
Karlskrona
Karlstad
Katrineholm
Kramfors
Kristianstad
Kristinehamn
Kumla
Kungsbacka
Kungälv
Köping
Laholm
Landskrona
Lidingö
Lidköping
Lindesberg
Linköping
Ljungby
Ludvika
Luleå
Lund
Lycksele
Lysekil
Malmö
Mariefred
Mariestad
Marstrand
Mjölby
Motala
Nacka
Nora
Norrköping
Norrtälje
Nybro
Nyköping
Nynäshamn
Nässjö
Oskarshamn
Oxelösund
Piteå
Ronneby
Sala
Sandviken
Sigtuna
Simrishamn
Skanör
Skara
Skellefteå
Skänninge
Skövde
Sollefteå
Solna
Stockholm
Strängnäs
Strömstad
Sundbyberg
Sundsvall
Säffle
Säter
Sävsjö
Söderhamn
Söderköping
Södertälje
Sölvesborg
Tidaholm
Torshälla
Tranas
Trelleborg
Trollhättan
Trosa
Uddevalla
Ulricehamn
Umeå
Uppsala
Vadstena
Varberg
Vaxholm
Vetlanda
Vimmerby
Vänersborg
Värnamo
Västervik
Västerås
Växjö
Ystad
Åmål
Ängelholm
Örebro
Öregrund
Örnsköldsvik
Östersund
Östhammar

A.2 Detailed results of the 500 random tests conducted and presented in figures 5.2, 5.3 and 5.4 sorted from the lowest to the highest response time.

No. of waypoints	No. of traffic situations	Response time
62	34	1454
20	0	1763
49	43	1930
29	26	1939
52	28	1948
86	134	1970
28	26	1985
40	0	2003
24	0	2019
51	4	2035
68	4	2045
45	0	2083
95	215	2089
75	28	2103
58	40	2106
67	66	2135
72	8	2135
38	39	2150
63	22	2171
18	24	2190
45	42	2190
47	7	2194
104	153	2195
41	40	2201
30	50	2203
68	45	2219
49	96	2223
62	39	2230
32	51	2237
43	75	2239
70	72	2247
52	66	2250
25	112	2252
16	104	2266
58	80	2273
52	57	2287
52	39	2290
87	105	2290
33	120	2295
77	62	2299
62	6	2302
74	50	2302
71	50	2312
60	51	2348
55	100	2351
50	19	2353
90	64	2354
86	50	2368
93	65	2382
100	103	2398
77	41	2409
69	71	2415
84	108	2421
70	102	2423
76	12	2431
36	104	2477
87	41	2481
103	48	2492
39	145	2498
91	85	2502
72	106	2520
50	124	2532
77	112	2539
89	169	2550
43	144	2551
65	124	2559
59	328	2586
80	141	2589
83	157	2597
94	191	2607
107	274	2613
108	144	2641
32	169	2642
72	198	2654
36	14	2656
63	156	2657
67	86	2664
70	198	2667
68	182	2680
73	95	2680
77	209	2681
61	90	2684
17	82	2699
104	144	2714
96	185	2715
90	40	2727
39	296	2735
92	157	2742
93	40	2749
105	169	2756
62	38	2766
33	295	2775
38	210	2775
109	140	2779
47	178	2785
78	88	2793
74	96	2794
85	480	2799
90	194	2809
96	104	2813
52	72	2825
79	193	2854
92	423	2860
86	222	2866
83	176	2872
96	252	2884
23	246	2885
22	399	2900
82	129	2912
116	138	2915
86	203	2916
55	218	2918
110	486	2925
79	229	2928
105	231	2929
66	324	2948
84	139	2970
76	192	2991
104	272	3003
51	378	3008
82	60	3011
116	215	3014
77	20	3015

No. of waypoints	No. of traffic situations	Response time
115	263	3025
46	318	3034
44	240	3050
51	378	3051
63	22	3054
92	277	3070
55	448	3076
83	311	3076
77	209	3081
124	297	3090
73	265	3108
74	495	3111
83	242	3121
116	290	3128
76	218	3145
56	319	3152
114	170	3156
114	140	3160
113	131	3163
73	298	3164
100	321	3177
60	500	3181
111	166	3190
94	359	3201
104	189	3207
76	302	3213
92	342	3214
82	213	3217
95	136	3217
78	329	3220
85	284	3237
88	443	3238
94	258	3241
95	187	3249
82	207	3253
120	360	3257
105	408	3261
91	386	3264
88	307	3268
119	304	3271
78	108	3275
102	569	3283
71	728	3284
118	201	3286

No. of waypoints	No. of traffic situations	Response time
92	342	3293
74	281	3311
93	212	3321
38	476	3332
93	72	3337
116	413	3339
117	410	3342
51	406	3353
102	432	3361
82	403	3362
81	373	3367
117	339	3367
97	142	3379
66	390	3380
113	372	3382
91	376	3392
88	128	3411
106	420	3412
64	461	3426
93	455	3427
147	418	3434
103	342	3436
106	282	3446
82	344	3457
69	394	3459
88	437	3460
60	426	3465
78	535	3472
129	204	3484
119	330	3485
37	414	3486
102	310	3488
94	123	3504
120	382	3505
167	424	3512
99	554	3513
82	429	3527
94	216	3548
79	329	3562
57	574	3567
85	521	3580
111	482	3582
77	586	3588
137	234	3613

No. of waypoints	No. of traffic situations	Response time
37	238	3622
120	382	3625
93	449	3630
144	467	3638
91	202	3655
58	673	3660
106	140	3660
104	474	3669
78	332	3683
106	402	3694
82	456	3695
128	379	3698
105	412	3709
96	502	3712
54	456	3716
51	591	3729
127	441	3743
125	282	3747
95	555	3769
48	640	3772
147	348	3773
50	402	3789
118	476	3814
120	549	3815
79	409	3833
78	484	3840
49	217	3864
118	249	3864
99	633	3866
103	290	3871
105	624	3873
38	210	3895
132	534	3914
125	293	3933
108	575	3948
105	553	3959
77	567	3963
79	760	3966
103	550	3969
85	596	3985
63	498	3998
123	361	4003
95	649	4020
97	246	4021

No. of waypoints	No. of traffic situations	Response time
108	479	4029
69	541	4039
98	553	4044
132	642	4058
137	526	4067
74	786	4070
65	796	4076
97	752	4119
97	568	4125
111	700	4127
120	581	4128
62	607	4156
114	470	4176
85	671	4185
101	631	4195
156	623	4215
102	565	4224
127	745	4239
94	466	4248
115	678	4262
50	604	4266
130	754	4274
107	638	4280
70	139	4287
99	592	4290
60	756	4296
113	697	4300
72	836	4306
103	622	4319
116	759	4320
134	615	4328
110	854	4345
135	633	4345
154	556	4347
93	648	4361
110	738	4364
119	594	4372
126	689	4376
96	701	4384
90	275	4396
98	611	4396
174	558	4430
137	817	4466
125	577	4485

No. of waypoints	No. of traffic situations	Response time
134	700	4498
69	972	4505
101	644	4535
181	872	4539
130	746	4543
76	723	4562
98	763	4582
102	833	4585
105	775	4589
73	934	4594
105	713	4594
76	818	4602
148	700	4603
148	960	4609
87	786	4614
92	683	4617
97	832	4621
112	868	4624
170	476	4670
63	865	4686
121	673	4699
122	820	4721
91	881	4729
89	770	4741
60	1005	4756
112	694	4773
136	669	4791
124	1005	4799
94	916	4823
95	651	4847
83	788	4850
85	1051	4857
68	954	4864
82	911	4873
95	621	4876
133	606	4882
165	714	4882
81	901	4914
76	464	4918
72	1189	4966
94	902	4996
46	750	5007
125	838	5009
119	997	5025

No. of waypoints	No. of traffic situations	Response time
64	717	5031
111	1021	5037
76	962	5056
144	145	5058
84	1048	5092
95	802	5122
106	1092	5127
100	932	5130
99	956	5132
90	872	5138
117	838	5142
141	972	5163
68	728	5190
90	1177	5194
152	1178	5234
161	1079	5245
123	1165	5275
117	788	5300
76	544	5333
170	1013	5354
113	1188	5377
124	1273	5394
63	1262	5415
79	1074	5436
61	1386	5449
115	589	5475
55	1218	5484
102	889	5501
75	591	5546
110	792	5548
114	1214	5548
171	922	5555
92	1358	5560
118	1101	5590
71	992	5604
87	1113	5608
101	972	5628
121	1274	5639
78	851	5641
139	996	5681
110	918	5720
89	930	5721
113	1321	5728
54	1107	5742

No. of waypoints	No. of traffic situations	Response time
108	1368	5745
108	947	5769
106	898	5790
148	753	5834
69	1368	5840
110	766	5854
78	1529	5861
97	572	5861
126	578	5877
131	1005	5903
41	1394	5905
77	1052	5909
108	1488	5933
138	364	5940
82	1596	5985
107	1282	6082
122	1408	6095
140	1160	6128
151	1388	6131
167	908	6131
182	1164	6134
114	998	6146
126	1053	6183
95	1359	6197
127	1206	6197
121	1364	6203
92	1525	6212
168	1336	6221
96	1466	6267
43	798	6272
49	1730	6324
135	1209	6327
132	1552	6387
85	793	6497
109	1502	6507
111	1485	6517
123	887	6517
83	1582	6528

No. of waypoints	No. of traffic situations	Response time
76	1566	6626
120	1088	6668
82	1666	6676
122	1264	6730
81	1525	6819
83	1662	6826
106	1646	6848
74	1614	6883
121	744	6926
141	1644	6978
70	1472	6986
167	751	7059
116	1352	7103
110	1765	7153
102	1571	7160
77	1693	7192
115	1352	7203
125	1640	7302
84	1716	7412
117	1679	7504
107	1746	7565
75	1985	7635
125	1996	7714
111	1497	7835
109	1701	7853
105	1984	7915
55	1746	7986
64	2104	8017
117	1688	8028
135	1995	8143
106	1800	8205
122	1989	8216
55	1964	8261
111	1763	8284
189	1739	8387
123	1938	8398
57	1250	8548
106	1823	8563

No. of waypoints	No. of traffic situations	Response time
78	2036	8694
186	1428	8717
99	1476	8786
60	2503	8836
86	2156	8849
60	2354	8973
140	2173	8989
108	2507	9051
134	2372	9068
116	2448	9150
106	1770	9184
98	2164	9237
94	1284	9295
110	1878	9519
149	2358	9542
156	2303	9725
129	2552	9887
139	2462	10203
117	2032	10378
113	2486	10450
78	2797	10542
92	2687	10846
163	2673	10924
88	2987	11094
120	2486	11165
68	1560	11331
182	2773	11389
79	2884	11435
125	2882	11463
114	2014	11534
105	2936	11641
167	2387	11650
108	3084	12169
86	2403	12933
110	2553	13234
136	2606	13723
140	3138	15332

TRITA -ICT-EX-2017:38