Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering

The design of a system for transmission condition analysis

Yevheniy Tomenchuk

Supervisor: Ing. Martin Košťál, MSc.
Supervisor–specialist: Jan Pospíšil
May 2021

Acknowledgements

First and foremost, I would like to thank my supervisor Ing. Martin Košťál, MSc. for his valuable advice and patient guidance. I am also grateful to my supervisor-specialist Jan Pospíšil, lead-developer at Eaton European Innovation Center in Prague, for lots of assistance, constructive comments and research materials. Many thanks to the EEIC for the internship work experience. Finally, I thank my family for the support during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating a final academic thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on 21.05.2021 .....

Abstract

The present thesis deals with the development of a client-server application based on the analysis of the existing component of the application(IntelliConnect) for vehicle faults evaluation and evaluation rules modification. The .NET Core framework, in pair with C# language, is used for the server-side of the application, while ReacJs is used for its client-side to achieve the set goals.

The elaborated client-server application has successfully eliminated the shortcomings of the existing system found during the analysis. The developed client application that communicates with the server using public API has substituted the method of changing evaluation rules within the OOXML tables and manual transfer. A new website provides a tool for modification and automated correctness checking of evaluation rules, simplifying their definition process. Moreover, it makes it possible to simulate vehicle faults, thus check the rules logic correctness. Having saved the original tool for importing rules in OOXML format, it has been advanced and expanded with the possibility to export evaluation rules in the same format. All manual processes connected with modifying rules and evaluating faults have been automated.

Keywords: CAN bus, J1939 protocol, Eaton, IntelliConnect, remote diagnostics, .NET Core, REST API, ReactJs

Supervisor: Ing. Martin Košťál, MSc.
Jugoslávských partyzánů 1580/3
Room A-526

Abstrakt

Daná práce se zabývá vývojem klient-server aplikace na základě analýzy stávající komponenty aplikace (IntelliConnect) pro hodnocení chybových hlašek vozidla a úpravou pravidel jejich hodnocení. .NET Core framework, ve dvojici s jazykem C#, se používá na straně serveru, zatímco ReacJs se používá na straně klienta k dosažení stanovených cílů. Vypracovaná klient-server aplikace úspěšně odstranila nedostatky nalezené během analýzy stávajícícho systému.

Vyvinutá klientská aplikace, která komunikuje se serverem pomocí veřejného API, nahradila metodu změny pravidel hodnocení v tabulkách OOXML a jejich manuální přenos. Nový web poskytuje nástroj pro úpravu a automatické ověřování správnosti pravidel hodnocení, což zjednodušuje jejich definiční proces. Kromě toho umožňuje simulovat chyby vozidla, a tak kontrolovat správnost logiky pravidel. Původní nástroj pro import pravidel ve formátu OOXML byl vylepšen a rozšířen o možnost exportovaní vyhodnocovacích pravidel ve stejném formátu. Všechny manuální procesy spojené s úpravami pravidel a vyhodnocením chyb byly automatizovány.

Klíčová slova: CAN sběrnice, J1939 protokol, Eaton, IntelliConnect, vzdálená diagnostika, .NET Core, REST API, ReactJs

Překlad názvu: Návrh systému pro analýzu stavu převodovky

Contents

Figures

Tables

Introduction

Motivation. The automotive industry has been consistently developing since the first car was invented by Karl Benz in 1886 [1]. In slightly more then a century the vehicle has evolved from an almost mechanical metal box with wheels into the car with a driver assistance system referred to as Autopilot like Tesla. Nowadays, automobile manufacturers are facing real and pressing challenges in the industry that has to react swiftly to the expectations of modern customers. The main concern has been shifted from just fast and well-designed cars to vehicles equipped with the continuously increasing amount of advanced technologies that are able to guarantee safety, uptime and efficiency. The integration of hardware and software into cars helps to improve their functionality, however, it requires additional analysis of the working order under different conditions and loads. As a result, the complexity of vehicles increases and it leads to product failures, which are hard to reduce and foresee. The development of advanced diagnostic tools might help to find some solutions to solve this problem. That accounts for the topicality of this thesis.

The bachelor work has been carried out under the supervision of the Eaton European Innovation Center (EEIC) whose specific focus areas include vehicle powertrains, industrial automation, power distribution, hydraulics, electronics and IT [2].

Problem statement. Due to the fact that the increasing number of electronic products are being incorporated on trucks, automobiles and other vehicles, the complexity of troubleshooting the product has become unmanageable without maintaining the computational diagnostic capability of on-board and off-board controllers.

In fact, several different diagnostic tools have been developed in the automotive industry. One of the tools Diagnostics 4.0 was provided by Softing, which offers remote and cloud diagnostics for vehicles with huge amounts of Electronic Control Units (ECU) [3, 4]. Another tool X-TRAILERPULSE was developed by Wabco [5]. It is trailer telematics solutions that capture a maximum of data from trailers with limited electronics, such as curtain siders [6]. Adding to this, Eaton provides IntelliConnect [7], that is a tool for remote diagnostics that provides near real-time monitoring of vehicle fault codes, prioritizes the critical events and provides accurate and comprehensive action

plans by technical experts that will increase uptime by reducing unplanned downtime and quicker repair diagnosis [7, 8].

The present thesis deals with the analysis of the existing component of the application(IntelliConnect) for evaluation of fault codes from the CAN bus and the design of an upgraded one. The new implementation makes it possible to integrate with other components and to modify the evaluation rules for technical staff.

Goals. The aim of the thesis involves performing the following specific tasks:

- to study works on the research topic that serve as a theoretical basis of the thesis;
- to examine and describe the main notions of the research;
- to analyze the existing solution (IntelliConnect);
- to substitute the current process of the OOXML table manual transfer and create a public Application Programming Interface (API), i.e. microservice;
- to develop a User Interface (UI);
- to simplify evaluation rule definitions (logical operators AND, OR and nested conditions) and verify their correctness (duplication, unavailability, etc.);
- to provide a dynamic view of fault codes (simulate a group of fault codes and calculate a given severity);
- to advance the import/export of evaluation rules in OOXML format;
- to allow a connection with new clients (HTTP endpoint).

Thesis outline. The thesis consists of the Introduction, 4 chapters, general conclusions, the references and the appendix.

The introduction presents the aim of this study, the research goals, focuses on the motivation and topicality as well as the problem statement.

Chapter 1 deals with the necessary theoretical basics and ideas on which this thesis is based.

Chapter 2 focuses on the analysis of the existing solution.

Chapter 3 offers a suitable architecture and design for the application. In addition, provides the practical part, where the development of the application is described.

In conclusion, the theoretical and practical results are summed up, the main goals of the study are substantiated as well as prospects for further investigation are outlined.

The appendix presents the acronyms used in the thesis.

Chapter 1

Preliminaries

The initial step of the thesis begins with the review of the Control Area Network, vehicle applications and truck diagnostics. It covers the necessary theoretical basics and ideas on which this thesis is based.

1.1 CAN - Control Area Network

One of the most crucial parts of the vehicle is Electronic Control Units (ECU). They can be defined as specific embedded systems that are accountable for handling electrical systems of the car such as windows, engine, brakes or doors etc [9].

It was not until the early 80s when electronic units started to equip vehicles [10]. There was a gradual rise in the demand for real-time data transmission within a vehicle. However, it was soon impossible to add more dedicated signal lines, due to costs, safety and repair issues. For instance, the cable length of some advanced European vehicles from the late 1980s was more than 2km [11]. To satisfy this pressing need for multiplexed communication, at the beginning the 80s the "Robert Bosch GmbH" company invented the CAN [11].

The Controller Area Network (CAN) is referred to as a multi-master, message broadcast system, i.e. a robust serial bus communication protocol [12]. It sets a standard for efficient and robust communication between devices, ECUs, actuators and other nodes in real-time applications within the CAN bus. Originally the aim was to make vehicles more fuel-efficient, secure and safe as well as to reduce wiring weight and complexity. The CAN protocol has obtained great popularity in the automotive industry since its appearance. Various trucks, automobiles, spacecraft, boats, ships and other vehicle types use it. Moreover, the protocol has widely been integrated into industrial automation and other fields of networked embedded control, being applied in different goods such as weaving machines, wheelchairs, medical equipment, building automation, and production machinery.

In the automotive industry, embedded control has developed from separate systems to networked and integrated control systems. Furthermore, introducing networks in vehicles enables to perform diagnostics more efficiently and to coordinate the operation of the stand-alone subsystems more accurately

(see Figure 1.1).

Figure 1.1: Vehicle ECUs network without CAN vs. with CAN

1.1.1 Description

The CAN specification determines the protocols for two layers of the OSI model: data link and physical (see Figure 1.2), that allow communication between the network nodes [13]. The application process of a node decides when the message frame should be transmitted. The message frame is made up of an identifier, data, and control fields. Considering the fact that the application processes are asynchronous, the CAN bus provides a conflict resolving mechanism.

Thus, the CAN protocol is included in the CSMA/CD class of protocols. The CSMA stands for Carrier Sense Multiple Access [14]. Every network node has to control the bus during the inactive period before transmitting a message (Carrier Sense). Furthermore, for the period of no activity, each network node has an equal opportunity to send a message (Multiple Access) [13]. The CD stands for Collision Detection [14]. In case two network nodes start sending messages simultaneously, the collision is detected and consequently, the appropriate action is taken.

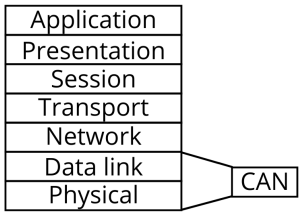

Figure 1.2: ISO/OSI Reference Model

Message formats. CAN distinguishes four messages also referred to as CAN frame types [15]:

- **Data frame** a frame that contains data for transmission;
- **Remote frame** a frame that requests the specific identifier transmission;
- **Error frame** a frame that signalizes the error occurrence detected by the node;

- **Overload frame** a frame that injects delay between either data or remote frames.

The standard data frame is presented in Figure 1.3. It begins with the Start of Frame (**SOF**) bit, which is used to synchronize transmissions on a CAN bus and is followed by an 11-bit **Identifier**. The latter sets the priority of the message(lower binary value means higher priority). Next comes the Remote Transmission Request (**RTR**) bit, which is set for remote frames. The **Control** field is a length code that represents how many bytes of data are transmitted. The **Data** field contains message payload of size up to 64 bits. The Cyclic Redundancy Checksum (**CRC**) contains a checksum that enables the receiver to detect the error occurrence. The Acknowledgement (**ACK**) field is sent by the receiver when accepting a valid message. The message ends with End of Frame (**EOF**), which signals the end of a CAN frame.

Figure 1.3: CAN message frame

In addition to the standard data frame, there is an extended one. At present, these are two formats defined in ISO 11989 standard – Standard CAN and Extended CAN [16]. They differ in the length of the identifier (11-bits and 29-bits, respectively). The identifier field performs the function of a descriptor of the data transmitted by the CAN frame.

1.1.2 Protocol extensions

The majority of network applications are based on the layered approach to system implementation. This layer-organized approach enables compatibility among products manufactured by different companies. The International Standards Organization (ISO) created a certain standard considered as a template to be followed within this systematic approach. It is referred to as the ISO Open Systems Interconnection (OSI) Network Layering Reference Model and is presented in Figure 1.2.

While the CAN protocol defines only the lowest two ISO/OSI model layers (see Figure 1.2), in numerous situations, it is necessary to use standardized protocols that specify the other higher layers. Existing higher-layer protocols are often adapted to a particular application domain. Examples of such protocols include SAE J1939, CANopen, DeviceNet and CANKingdom.

SAE J1939. The one that is specially developed for vehicle applications is SAE J1939. In the last few years, efforts have been made to join CAN and Ethernet, which is the leading technology for local area networks (LAN) as well as the common connecting source to the Internet. J1939 protocol was

developed by the Society of Automotive Engineers [17] for the automotive industry purposes to define the higher-layer communication control. It is a significant advantage to have a standard because the latter allows autonomous development of the separate networked components, which leads to the compatibility of components for vehicle manufactures provided by different suppliers. For example, the J1939 protocol determines rules for reading and writing data as well as calibrating certain subsystems. K. Etscherberger states that the data rate of SAE J1939 is about 250 kbps, which gives up to about 1850 messages per second [18]. J1939 protocol has got a wide range of applications like truck communication, machinery in forestry and agriculture, marine navigation systems etc.

CANopen. Another standardized application is CANopen, which is on top of CAN and is used in industrial automation across Europe. The standard was specified by CAN in Automation (CiA), the international organization that develops and supports CAN-based higher-layer protocols [19]. CANopen defines device and communication profiles, allowing independent use of CAN. While the communication profile determines the fundamental communication mechanism, the device profile is used by the common devices in industrial automation, e.g. digital and encoders, controllers and analogue I/O components. The device profile can be configured autonomously of its manufacturer by means of CANopen, which differentiates between real-time and less critical data exchange. CAN open provides standardized communication objects for various data(real-time, configuration, network management) as well as certain special functions (timestamp and synchronization messages).

DeviceNet. Another standardized application is DeviceNet, which is primarily used for distributed industrial automation in the U.S.A and Asia. It was originally developed by Rockwell Automation [20]. DeviceNet, as well as Transmission Control Protocol/Internet Protocol (TCP/IP), are defined as open network technologies. Sharing upper layers of the communication protocol they are based on lower layers. While DeviceNet is built on top of CAN, TCP/IP is built on top of Ethernet.

CANKingdom. One more standardized high-layer protocol based on CAN is CANKingdom provided by Kvaser [21]. It is used for motion control systems. By means of this protocol network behaviour can be changed anytime, even while the system is operating. For example, when a failure occurs CANKingdom makes it possible to turn off individual network nodes. The CAN node identifiers and message sending triggers can be altered while the system is operating. Real-time network reconfiguration is used when a failure occurs, e.g. a failure of a radio link ECU in a marine application. The network monitor, also referred to as the King, in that situation firstly turns off the radio node to prevent it from sending any further commands, and secondly instructs the relevant nodes to obtain data from the King. Thus, the problem of a node receiving two simultaneous but conflicting commands and the problem of two nodes sending the same CAN id are eliminated [22].

Having analyzed the higher-layers protocols it becomes obvious that they have been developed for different purposes and with different applications taking into account various demands. It is reflected, for instance, in their real-time control support. Though J1939 protocol is used for control algorithms implementation, it does not support time-limit communication. As opposed to CANKingdom and CANopen that implements such functionalities and provide inter-node synchronization support. Moreover, CANKingdom and CANopen enable static and dynamic network configuration, whereas the J1939 protocol has less flexibility.

1.1.3 A brief history

The evolution of microelectronics has paved the way for the appearance of distributed control systems in vehicles. However, there was no standardized and low-cost protocol that was appropriate for real-time control systems in the early 1980s. Therefore, the development of a new serial bus system was critically important and started at Bosch in 1983 [23]. As a result, CAN was launched in 1986 at the SAE Congress in Detroit [24]. The creation of CAN was primarily driven by the need for new functionalities, but the need for wiring was also considerably reduced (see Figure 1.1). The Bosch CAN Specification 2.0 was released in 1991 [25] and the CAN protocol was then standardized internationally as ISO 11898-1 in 1993 [26]. There was early awareness of the need for higher-layer protocols. In 1991, Kvaser launched CANKingdom [10]. Another higher-layer protocol DeviceNet was defined by Allen-Bradley in 1994 [27]. The subsequent protocol CANopen was presented by CAN in Automation (CiA) in 1995 [28]. Initially, CAN was applied only for engine control, but nowadays there is a wide spectrum of applications for chassis and powertrain control as well as for body electronics and infotainment systems.

1.2 Standardized vehicle diagnostics

As it was outlined in the previous section, nowadays the majority of vehicles use the CAN for ECU communication. However, the CAN can be compared to a "telephone" that serves as an instrument for communication while the "language" for conversation is not provided. The solution appears and the standardized CAN communication method("language") for vehicles, as well as the principles for the exchange of the information and diagnostic data between ECUs, are described in J1939 protocol developed by SAE. But originally the protocol was developed for vehicles built after 1985 and it was the J1587 [29]. Starting from 2007, it was displaced in favour of the new and upgraded J1939 protocol to make the most of the CAN features in modern multi-ECU vehicles.

J1939 is a highly advanced protocol that requires the specific message's format, thus it takes advantage of the Extended CAN Identifier (29-bit) [30].

J1939 frame identifier contains the following parts (see Figure 1.4):

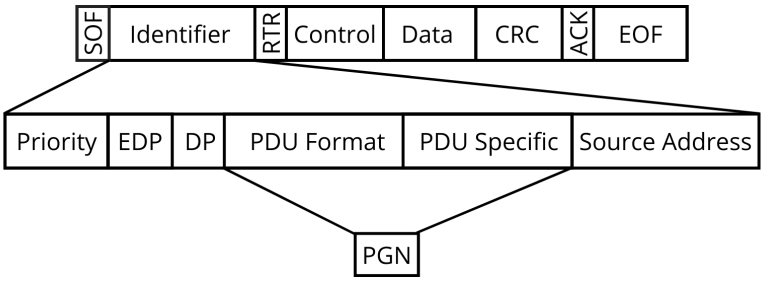

Figure 1.4: J1939 29-Bit message identifier

- Priority - establishes the message priority, the highest priority being 0 and the lowest is 7;
- EDP - Extended Data Page together with DP differentiates message definitions;
- DP - Data Page together with EDP differentiates message definitions;
- PDU Format - Protocol Data Unit Format determines the parameter group the message belongs to and distinguishes the type of data;
- PDU Specific - Protocol Data Unit Specific defines the message's destination ECU address;
- Source Address - defines the message's source ECU address.

The Parameter Group Number (PGN) is the concatenation of PDU Format and PDU Specific (see Figure 1.4), used to differentiate between types of messages. Parameter Groups and their numbers (PGN) are listed in SAE J1939 and defined in SAE J1939/71 [31]. Another lookup table of ISO CAN Bus standard items can be found on the web [32].

1.2.1 Diagnostic message

In J1939 diagnostic data are represented as Diagnostic Messages(DM), that are used to monitor, test and clear diagnostic information in devices on the network. There are 19 different diagnostic messages defined in J1939. The examples of some of them are described beneath:

- DM1 Message provides a list of Diagnostic Trouble Codes (DTC) that report the diagnostic condition of the vehicle ECU over the CAN network. These DTCs are currently active on the device and are referred to as Active Diagnostic Trouble Codes [33];
- DM2 Message provides the list of DTCs that report the diagnostic condition of the vehicle ECU over the CAN network with the same details. These DTCs are not currently active but were active at some time in the past and are referred to as Previously Active Diagnostic Trouble Codes [33];

- DM3 Message is the message indicating that all the diagnostic information concerning the Previously Active DTCs should be cleared or Inactive DTCs should be reset. This guarantees that the Active DTCs present in ECU are not impacted. This message is referred to as Diagnostics Data Clear [33].

1.2.2 Diagnostic Trouble Code

The Diagnostic Trouble Code (DTC) also sometimes referred to as Fault Code is not a single value or code, but a set of information which is used to specify a problem in the device [34]. The DTC structure consists of the following parts:

- Suspect Parameter Number (SPN) defines the source of the issue. An SPN is assigned to each J1939 parameter [35].
- Failure Mode Indicator (FMI) determines the data parameter that is the type of the issue [36].
- Occurrence Count (OC) defines the data parameter that indicates the number of times the issue has occurred;
- Conversion Method (CM) specifies how the SPN and FMI should be handled or translated, primarily used to deal with older diagnostic protocols.

1.3 Vehicle Diagnostics by Eaton

Eaton is one of the world's leading producers of automated, constant mesh and synchronized manual transmissions [37]. Original vehicle manufacturers all over the world count on Eaton's solutions that improve the overall vehicle drivability, efficiency, reliability and safety. Eaton offers a wide spectrum of transmissions, that were created with different applications in mind for different purposes. The Eaton Transmission Families are listed below:

- Endurant [38];
- Fuller Advantage [39];
- UltraShift Plus [40];
- Procision [41];
- AutoShift / UltraShift [42].

As it was mentioned before, Eaton just not only creates vehicle commercial components but also provides a suite of connected solutions designed to enable their products to perform at their optimal capacity.

1.3.1 Eaton IntelliConnect™

Taking into account the fact that nowadays many trucks worldwide are hitting the road to places of their destinations, someone should take responsibilities for securing drivers' safety and maximizing vehicles' uptime. That's why fleet owners much depend on telematics systems, which provide important data for monitoring the current state of the vehicles, e.g. GPS, sensor and fault code data. But it is not an easy task to understand collected data when you have several fault codes for multiple vehicles or per one vehicle. Consequently, this accumulation of fault codes becomes a burning issue, which IntelliConnect by Eaton [7] can help to settle by providing expert guidance via remote diagnostics.

IntelliConnect is the platform that Eaton is building up for connected solutions. The key piece of this platform is a remote diagnostics of Eaton automated transmissions. It prioritizes the critical events and provides accurate and comprehensive action plans by technical experts, that will increase vehicle uptime by reducing unplanned downtime and quicker repair diagnosis [8].

Let's compare the vehicle fault life cycles with and without IntelliConnect.

The fault life cycle without IntelliConnect is shown in Figure 1.5. Initially, transmission sets a fault code, which is afterwards transmitted to the existing telematics provider. From there, the fault code is sent directly to the fleet with no analysis or immediate, actionable subsequent steps.

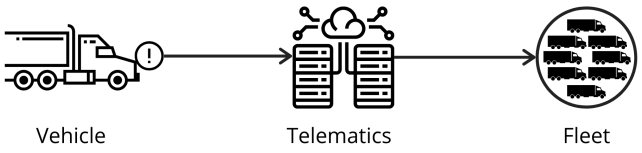

Figure 1.5: Vehicle fault code life cycle without IntelliConnect

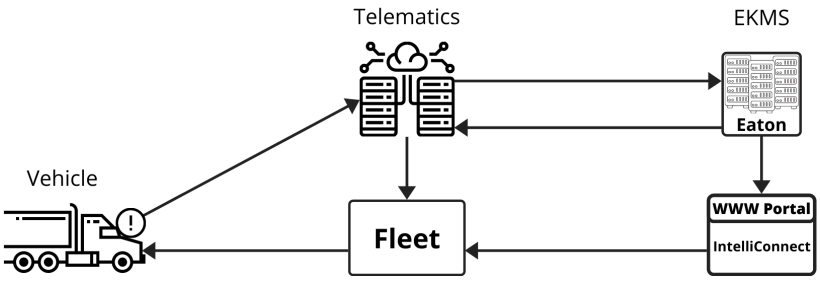

Figure 1.6: Vehicle fault code life cycle with IntelliConnect

The fault life cycle with IntelliConnect is shown in Figure 1.6. As soon as the transmission sets a fault code, it is sent to the telematics company. But now instead of being sent directly to the fleet, the fault code data are sent to the Eaton knowledge management system. IntelliConnect monitors all the fault codes of the vehicle, determines the ones that impact the driveline

and ranks them by severity. It analyzes the code data and provides a fault code action plan [43]. Further on the processed action plan is sent back to the telematics company and uploaded to the Eaton IntelliConnect portal site [44], where it can be accessed by the fleet owner. Finally, the fleet owner sets the recommended action plan to the driver.

Using a comprehensive systems approach, IntelliConnect takes into account data from other vehicle components to trace potential root causes and suggest solutions. It doesn't focus solely on transmissions, but it attempts to evaluate the overall state/health of a vehicle.

Chapter 2

Analysis

As mentioned in the previous chapter, the rules for faults evaluation are currently passing and modifying with OOXML tables by Eaton specialists. This work aims to optimize this manual process using the Rest API application for editing and debugging evaluation rules with UI on top of it. This chapter seeks to analyze the current state.

2.1 AS-IS vs TO-BE State

AS-IS. Currently, Eaton specialists edit rules in OOXML files and update the database using a simple console application to import it. They cannot automatically check the validity of changes made in evaluation rules, except for checking their accuracy manually. The human factor can easily lead to mistakes in rules. Furthermore, changing rules priority is really hard because they should manually specify each rule's unique priority in the rule group.

TO-BE. The final application should enable users to provide all needed operations with rules and automate all manual processes like changing rules priority and indicating invalid operations and data. Moreover, the system should enable the user to generate Service Activity Report request and evaluate it to check the rules logic flow.

2.2 Basic definitions of the analysis

This section explains all definitions that are needed to further analysis.

2.2.1 Service Activity Report

It is Eaton's specific request format for transferring data to be evaluated. It contains several blocks of data (see figure 2.1), including a vehicle with a list of its components and a list of faults, which are in the focus of the analysis since every evaluation rule is based on either a component or fault data.

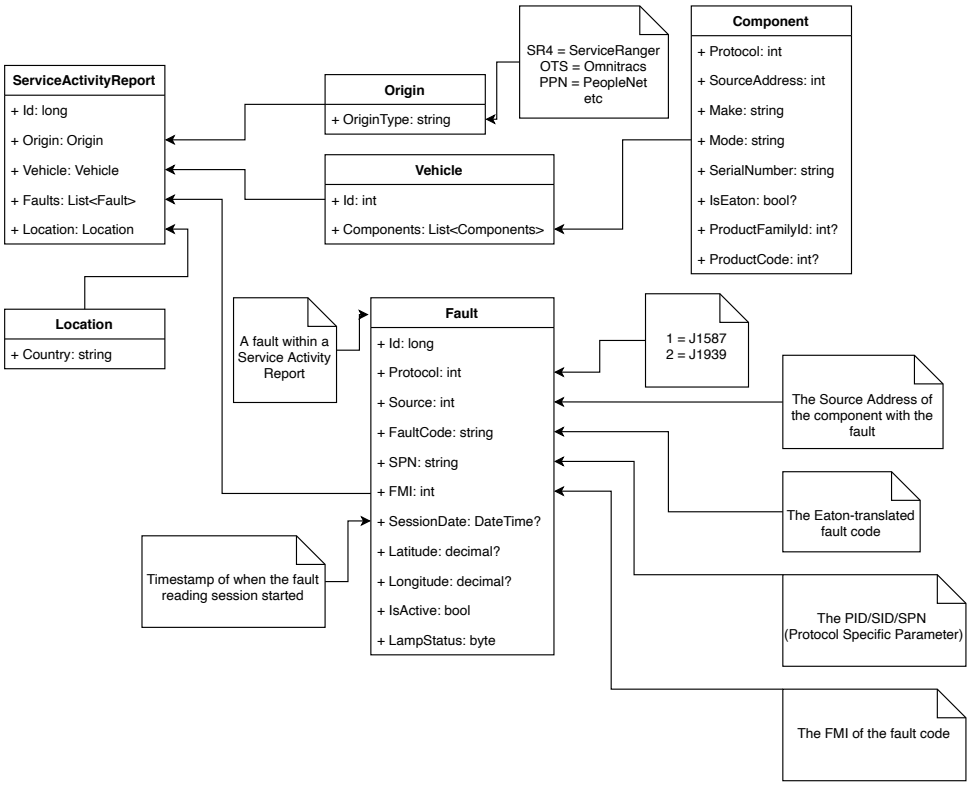

Figure 2.1: Service Activity Report Entity Model

2.2.2 Evaluation rule

An evaluation rule (see figure 2.2) is a group of data used for SAR evaluation that can be divided into the following sub-groups:

- **Rule Group:** this is the name of the rule group to which the rule belongs.
- **Rule logic:** these are the data about the rule **priority** number in the rules hierarchy, **prefix** and **suffix** used for writing complex rules for doing logical operations with other rules.
- **Origin Type:** this is a Service Activity Report origin type used to identify when an evaluation request comes from the predefined sources which are not supposed to be dealt with.
- **Vehicle component data:** these are the numbers representing **component source address**, **product family id**, **product code** and the indicator defining Eaton production component.
- **Fault data:** these are the numbers representing **fault source address**, **fault code**, **FMI**, **SPN** and the indicator defining the active state of the fault.

- **Evaluation result data:** this is a combination of two parameters. The **result type** indicator can be either "Evaluate" or "Action Plan". If it is "Evaluate", the **result key** will refer to the next rule group for further evaluation. If it is an "Action Plan", the evaluation process is finished with a definite action plan key.

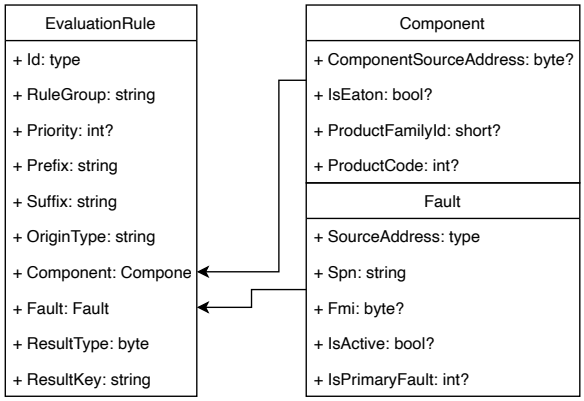

Figure 2.2: Evaluation Rule Entity Model

Three types of evaluation rules exist:

- **Root:** the rule contains component data if the result type is "Evaluate" and does not include fault data.
- **Non-complex/single:** the rule contains no component data but does have fault data.
- **Complex/multiple:** the set of rules that contain no component data but have fault data and create a complex expression using prefix and suffix.

2.2.3 Rule group

Rule group is the set of evaluation rules with the same rule group parameter. There are three different types of rule groups:

- **Root evaluation:** the leading rule group that points to single evaluation rule groups using result key parameter.
- **Single evaluation:** the rule group that goes after root evaluation contains non-complex rules and points to the multiple evaluation rule group.
- **Multiple evaluation:** this is the final rule group that contains a set of complex evaluation rules.

2.3 A typical cycle of using the application

Initially, the database is empty, and the user does not have an account. For using the application, the user creates a new account with a username and a password. The user authenticates in the application using his credentials. After the authentication and authorization process, the user sees the page with an empty table of evaluation rules. The user can initialize a new empty root rule group or import the existing one from the OOXML file. After initializing the root, the user can add, update, delete evaluation rules and change their priority by dragging and dropping table rows.

Furthermore, the user can import or initialize a new single rule group in the root. The new root evaluation rule that points to the newly created rule group will be added to the root evaluation rule group. Every evaluation rule with the result type "Evaluate" points to the rule group with the same name as the rule's result key.

In addition, the user can switch between different rule groups by clicking the open button that appears in the rules with the result type "Evaluate" in a table. The latter has a specific structure, and its contents dynamically change according to the type of a rule group.

Finally, the user can also opt to create a SAR request and evaluate it to check evaluation rules validity.

2.4 Requirements

The requirements are of two types:

2.4.1 Functional requirements

1. **Sign-in:** The user must be able to sign in to the system using a username and a password.
2. **Sign-in validation:** The system must be able to validate the user's sign-in input data.
3. **Sign-up:** The user must be able to create a new account with a username and a password.
4. **Sign-up validation:** The system must be able to validate the sign-up input data.
5. **Sign out:** The user must be able to log out from the account.
6. **Read evaluation rules:** The user must be able to read all evaluation rules within the same rule group.
7. **Create evaluation rules:** The user must be able to create new evaluation rules within the same rule group.

8. **Import evaluation rules:** The user must be able to import new evaluation rules to the same rule group using the OOXML file.
9. **Update evaluation rules:** The user must be able to update evaluation rules within the same rule group.
10. **Delete evaluation rules:** The user must be able to delete evaluation rules within the same rule group.
11. **Create rule group:** The user must be able to initialize a new rule group or import an existing rule group from the OOXML file.
12. **Delete rule group:** The user must be able to delete the whole rule group.
13. **Evaluation rules validation:** The system must validate any evaluation rules changes within the same rule group. Every evaluation rule has a different format depends on the rule group type.
14. **Evaluation rules priority:** The user must be able to change evaluation rule priorities within the same rule group using drag and drop. Priority must be changed automatically by the system.
15. **Export evaluation rules:** The user must be able to export evaluation rules with the same rule group as an OOXML file.
16. **Debug evaluation rules:** The user should be able to debug the evaluation rules logic by creating and evaluating a Service Activity Report request.
17. **Optimistic database concurrency:** The systems must detect all invalid or unexpected changes. Multiple users cannot edit the same rule group simultaneously.

2.4.2 Non-functional requirements

1. **ASP.NET Core or ASP.NET:** Server side of the application must be implemented using ASP.NET Core or ASP.NET framework in C language.
2. **Client UI:** The client's application must have a simple and user-friendly UI.

2.5 Use-case model

The use-case model (see figure 2.3) defines and clarifies functional requirements from the previous section.

Actors. The application has only one role, that is, the user.

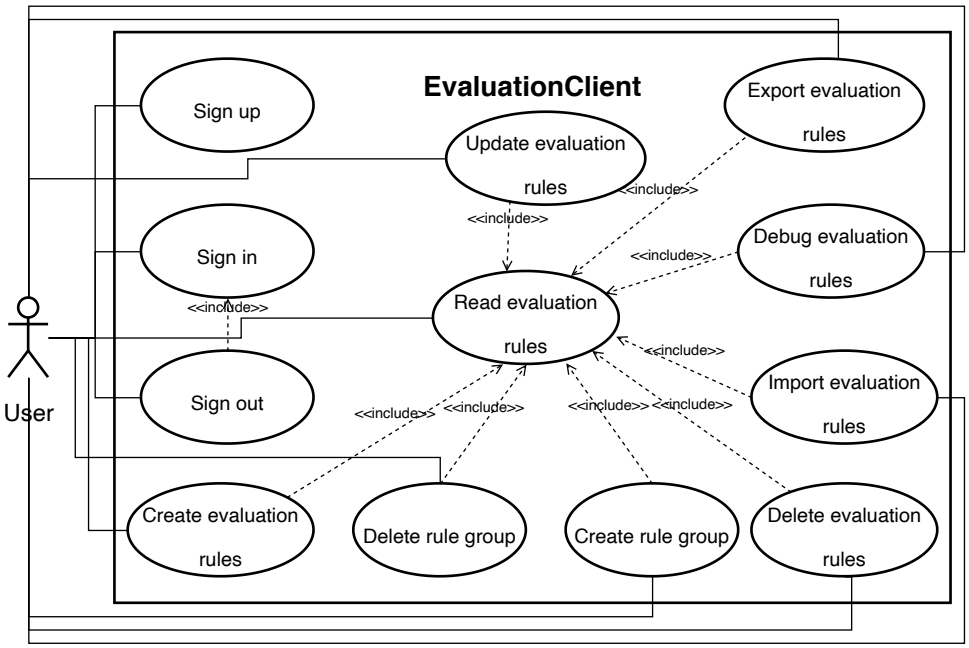

Figure 2.3: Use-Case Diagram

UC1: Sign-in. Use-case that enables the user to sign in to the system using a username and a password.

The main scenario: Classic sign in

1. The user enters a username and a password.
2. The system checks the validity of the username and the password and if the data are valid, authorize the user.

The secondary scenario: First sign in

1. It starts from step 2 of the main scenario, where the system successfully authenticates the user for the first time.
2. The user is directed to the main page of the application with the evaluation rules table.

UC2: Sign-up. Use-case that enables the user to create a new account for further authentication.

The main scenario: Classic sign up

1. The user enters a username and a password.
2. The system checks the validity of the username and the password.
3. The user is directed to the sign-in page for further authentication.

The secondary scenario: Alternative sign up

1. It starts from step 2 of the main scenario, where the system has checked the validity of the input data.
2. The user already exists, and the system warns the user.

UC3: Sign-out. Use-case enables the user to sign out from the account. After the sign-out process, the user is directed to the sign-in page.

UC4: Read evaluation rules. Use-case enables the user to read evaluation rules in the table, changing dynamically in accordance with the rule group type.

UC5: Create evaluation rules. Use-case enables the user to create new evaluation rules. The evaluation rule type changes depending on the rule group type. The system will validate all changes and estimate their impact on the current rule group.

UC6: Import evaluation rules. Use-case enables the user to import evaluation rules from the OOXML file with a predefined format. A temporary rule that points to the new rule group will be added to the current rule group. The system's ability to import a rule group depends on the current rule group. The system validates imported evaluation rules and their impact on the current rule group.

UC7: Update evaluation rules. Use-case enables the user to change evaluation rules data. The system will detect all changes and disable the saving option while data are invalid. The rule's priority can be changed by dragging and dropping rules in the table.

UC8: Delete evaluation rules. Use-case enables the user to delete evaluation rules. The system will validate the impact of the deletion on the current rule group.

UC9: Create a rule group. Use-case enables the user to create a new rule group. The temporary rule that points to the newly created rule group will be added to the current rule group. The system will validate the impact of creation on the current rule group.

UC10: Delete the rule group. Use-case enables the user to delete the rule group. The user must have a list of all possible rule groups to be deleted.

UC12: Evaluation rules priority. Use-case enables the user to change rules priority by dragging and dropping rules within table context. The system will change data automatically depending on the current rule group type.

UC13: Export evaluation rules. Use-case enables the user to export the rule group as the OOXML file with a predefined format. The user must have a list of all possible rule groups to be exported.

UC14: Debug evaluation rules. Use-case enables the user to create and evaluate Service Activity Report.

The main scenario: Debug

1. The user adds faults and components data to be evaluated.
2. The system validates the SAR request.
3. The system evaluates the request if SAR is valid.

Chapter 3

Design and implementation

3.1 Network architecture

Different architectures for designing a network are available. They specify network's physical components, operational principles and procedures, functional organization and configuration and communication protocols. In the bachelor thesis, two of them have been analyzed: **Peer-to-Peer** and **Client-Server** architectures (see figure 3.1). Though both Client-Server and Peer-to-Peer approaches have their pros and cons and one is not better than other, the Client-Server architecture provides us with certain advantages over the Peer-to-Peer model. That is why it has been chosen for the thesis.

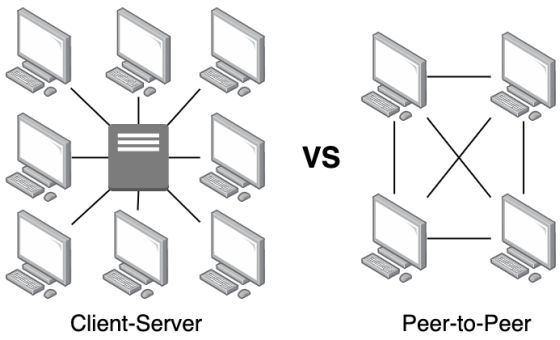

Figure 3.1: Client-Server vs Peer-to-Peer

The Client-Server model offers easier maintenance, security, and administration. As all data are stored on servers, they are more secure. Access is controlled by servers ensuring that only screened clients can use the application and change the data. One of the Client-Server model main advantages is centralization. i.e. all data are gathered in a single location. As a result, the data are well protected, and any problem occurring in the entire network can be solved in one place. While in P2P models, updates must be applied and copied to network's peers, which requires a lot of work and leads to errors. Furthermore, the updating data process has become more manageable in the Client-Server model. It is easier to authorize users of the network by requiring credentials like the username and the password. Client-Server networks are

also highly scalable, i.e. it is possible to expand the network by adding new nodes or to improve the server-side without any interruptions. Thanks to the centralization, even when the network is expanded, no issues with resources arise. Therefore, a small amount of stuff is required for the network configurations. Thanks to encapsulation it is possible to repair, upgrade, or replace servers without clients being involved. Encapsulation is the process by which an object can hide its data and methods without revealing them to users [45]. Finally, regardless of the platform or the location, each client is able to sign into the system.

On the other hand, there are some possible disadvantages. Firstly, in the Client-Server network with frequent simultaneous client requests. The server gets severely overloaded, which leads to traffic congestion. In P2P models, however, it is not an issue, because network resources are strictly proportional to the quantity of peers in the network. However, it is a minor issue for the project under research because Eaton's evaluation rules are hardly ever changed. Secondly, if a critical server error occurs, it will not process the client's requests due to its centralized character. Moreover, Client-Server architecture is short of the robustness of the P2P paradigm. Robustness refers to a network's ability to bounce back or continue functioning if one of the components fails [45]. Once in the Client-Server models a server fails, the requests fail too. Whereas in P2P networks, if a node fails or abandons the request, other nodes are not effected and can successfully finish their operations.

After considering the advantages and disadvantages of the Client-Server architecture and functional requirements described in the previous section, it has been decided to use this architecture in the thesis.

3.2 Using technologies

This section describes the technologies used for the application implementation, their advantages and disadvantages and explains the choices.

3.2.1 Back-end(Server)

Two options have been considered for the back-end, known as the server-side of the application: **ASP.NET Core** [46] and **ASP.NET** [47]. ASP.NET Core is a brand new, rewritten, and modern replacement of the ASP.NET Framework. Being a cross-platform framework, it can work on multiple platforms such as Windows, Linux and macOS, which is not supported by ASP.NET Framework. ASP.NET Core is open-source, while the ASP.NET framework is not. Moreover, ASP.NET Core is faster, lighter, more modular, scalable and designed to work with modern libraries and languages. For example, Entity Framework Core has better mappings, migrations, and query performance than the Entity Framework used in ASP.NET Framework. While both frameworks support WPF and Windows Forms to build modern Windows client applications, ASP.NET Core is still immature and under development.

ASP.NET Core lacks the functionalities present in the ASP.NET framework so far.

All things considered, both frameworks can be successfully used for building applications. However, much depends on the goals set for the developer. If a brand new application is being built and ASP.NET Core provides all the functionalities needed, it is a more promising option, as it is continuously improved and updated. Concerning existing applications using some libraries that ASP.NET Core does not yet support, it is much more reasonable to continue using ASP.NET Framework. Thus, if there is an existing application in the ASP.NET framework, there is no need to migrate it to ASP.NET Core.

3.2.2 Front-end(Client)

Two options have been considered for the front-end, known as the client-side of the application: **ReactJs** [48] and **VueJs** [49]. Both frameworks have got some advantages and drawbacks. ReacJs offers maximum flexibility and responsiveness. However, due to the complicated setup process, functions, properties and structure, extensive knowledge is needed to develop an application. At the same time, VueJs offers detailed documentation that can easily be learnt and basic knowledge of HTML and JavaScript is enough to do the job. Thus, VueJs is easier to use and upgrade an existing application. This framework occupies much less space and operates much better than others. Without the whole system being affected, Vue.js offers easier integration of smaller parts, both in a complex web interface and a single-page application.

Furthermore, the document object model form the basis of ReactJs and allows the browser-friendly management of documents in HTML, XML or XHTML formats. Vue.js embraces only standard HTML-based templates. Due to its scalability and flexible structure ReactJs is much better for large-scale apps. By contrast, while using VueJs for large-scale apps some development issues may occur. React is supported by professional developers who continuously search for possible improvements by putting in the effort to add more features. Even though VueJs elaborate design and architecture make it a leading JavaScript framework, it still has a minimal market share compared to React.

ReactJs and Vue.js can reach milestones in the development of the web applications thanks to their considerable flexibility, top speed, and modern features. However, considering the project's functional requirements, the first one has been chosen as more appropriate for the goals to be achieved.

3.2.3 Database

Two options of the database have been considered to be used in the project, namely **PostgreSQL** [50] and **Microsoft SQL Server** [51]. Both are very popular, but PostgreSQL is cheaper than MSSQL. Organizations with a tight budget would rather select PostgreSQL's interface. As one of the first database systems ever developed, it is often used for the creation of web databases allowing users to deal with either structured or unstructured data.

Being able to handle terabytes of data PostgreSQL is a scalable database management engine that supports JSON. It has got a wide range of predefined functions, and several interfaces are available.

However, some minor flaws have been identified. Firstly, the configuration appears to be somewhat confusing. Moreover, spotty documentation can be difficult to deal with. Finally, while processing the large bulk of operations or reading queries, it is not easy to maintain the high speed. contrast, MSSQL is very fast and stable. Its engine's ability to track and adjust levels of performance leads to resource reduction. This database engine works on both cloud-based and local servers, with the possibility to work simultaneously. Besides, the latest version allows dynamic data masking, guaranteeing the access of authorized individuals to sensitive data.

Many organizations can not afford the MSSQL, because the enterprise pricing is too high. Moreover, MSSQL is over-resource consuming, despite the performance tuning. Nevertheless, for large organizations with enough resources that use several Microsoft products, it is the best option.

3.2.4 Summing-up

Following the requirements of the company for which the application has been developed I had to chose from limited options of back-end, front-end and database, despite the fact that there far more existing ones. For implementing the server-side of the application, ASP.NET Core with relational database MSSQL has been used. The client-side of the application is developing with ReactJs and Semantic-UI-React [52]. The latter is the framework for creating responsive web pages using HTML attribute classes. To deal with a relational database the Oriented Relational Mapping (ORM) has been used by applying the Entity Framework Core predetermined by the choice of ASP.NET Core for the server. This technology enables mapping from Object-Oriented Programming to relational database tables. For requesting and manipulating the database data, Language Integrated Query (LINQ) is used.

All technologies used in the current implementation are part of the .NET Core. As a result, all code on the server-side is written in C#.

3.3 Components

The whole system has a multi-layer architecture and is divided into five components: **REST API**, **Business logic** and **Repository**, placed on the server, **EvaluationClient**, a web page in the browser on the client-side and **EvaluationDb**. All components are shown in figure 3.2.

EvaluationDb. As was mentioned in the previous section, the relational database MSSQL has been chosen, which is accessed by using an objective-oriented mapping from Entity Framework Core.

All database tables are generated from entity classes with code first strategy that works in the following way. Firstly, entity classes with proper annotations are created and, finally, the framework is left to generate tables.

Persistence. This component represents the application's data layer using ORM (Object Relational Mapping) and enables access to the relational database MSSQL. The technology used in this component is ASP.NET Core with Entity Framework Core.

Business logic. This component contains all the business logic of the application. It is dependent on the Persistence component, which provides access to the database.

REST API. This component represents the presentation layer of the server implemented as a simple REST API with ASP.NET Core framework and relies on the use of the Service component.

EvaluationClient. This component contains UI implemented with ReactJs in collaboration with Semantic-UI-React. It represents the client-side of the application and depends on the communication with the server using REST API controllers defined in the Controller component.

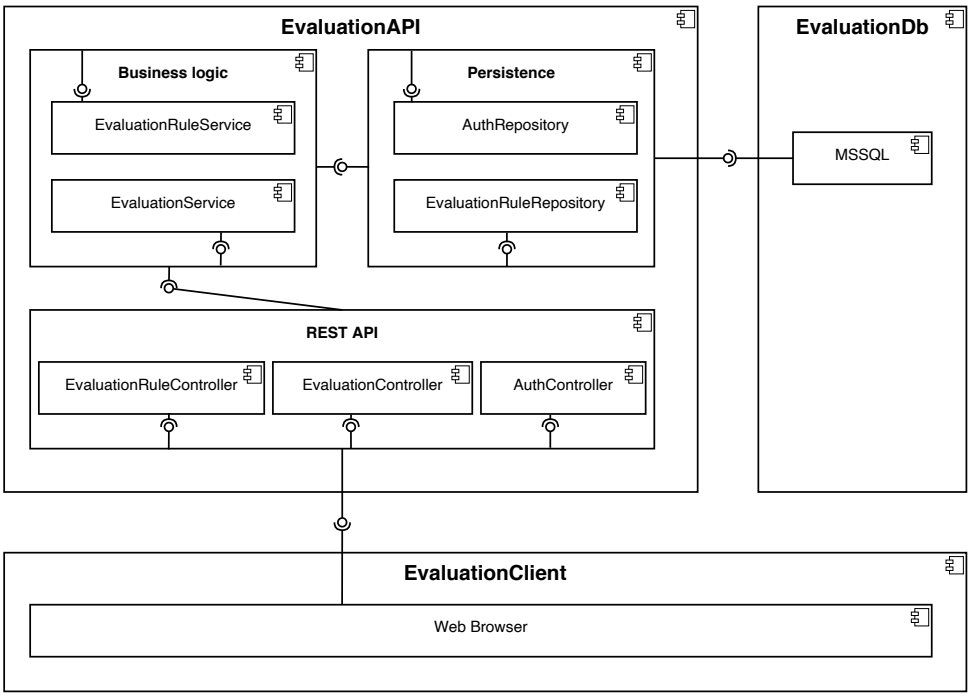

Figure 3.2: Component Diagram

3.4 Development environment

The vast majority of server-side code is written in programming language C# using ASP.NET Core. Visual Studio Enterprise Edition [53] has been chosen as a primary IDE for the server-side of the application. For working with database, Microsoft SQL Server Management Studio [54] has been used. As far as the client-side of the application is written in JavaScript language using ReactJs framework, it has been decided to use Intellij Idea Webstorm [55] as IDE.

3.5 Project structure

The first task of the implementation is to create a project with a readable structure. The latter is based on the design of the components defined in the previous chapter, i. e. each component has its folder or project.

- **Repository** - the data/persistence layer.
- **Service** - business logic.
- **Controller** - REST API controllers.
- **EvaluationClient** - client-side of the application.

Finally, models and data transfer objects for REST API communication are added.

3.6 Configuration management

For configuration management, **Azure DevOps** [56] is used, which provides a Version Control System (VCS). One programmer has written the project, and there is no need to follow any rules while working with VCS, but the situation can change in the future, and Azure DevOps can easily handle this.

```
public class MyClass {
    private Dependency dependency_;

    public MyClass(Dependency
       dependency) {
        dependency_ = dependency;
    }
}
```

Figure 3.3: Use of the dependency injection

3.7 Dependency injection

Dependency injection is a pattern that enables the inversion of control between class and his dependencies. As a dependency, any object needed to the class for its operation is considered.

```
void ConfigurationService(IServiceCollection s) {
    s.AddScope<Dependency>();
}
```

Figure 3.4: Configuration of the dependency injection

While creating a class using dependency injection, the dependencies are defined in the class constructor. After instances of dependency are specified. All dependencies are defined as parameters of the constructor, and then dependency injection injects required instances while creating the class instance. The use of this pattern is shown in figure 3.3 and configuration example is shown in figure 3.4.

Dependency injection is used almost everywhere in the project, and .NET Core supports it perfectly.

3.8 Layers realization

The architecture (see figure 3.5) and design described above have a goal to make the system easy extendable, and testable. The following section describes the steps of the implementation of different layers, patterns, and techniques used.

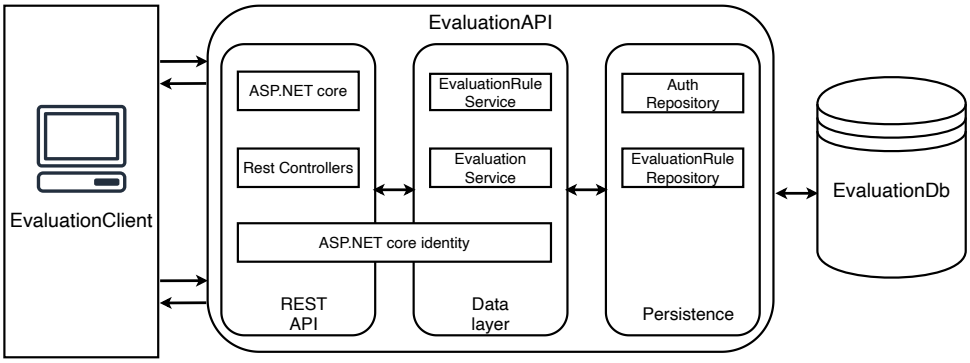

Figure 3.5: Architecture

Data/Persistence layer. The data layer is located in the "Repository" folder of the project and enables access to the database named as **EvaluationDb**. The only layer that has access to the database by using Entity Framework Core.

For every module of the application, the *NameRepository* is created, where the *Name* refers to the specific part of the application. For example, the evaluation rules database maps to *EvaluationRuleRepository*. For every repository, the data layer dependency injection is configured.

Business logic layer. The business logic layer is located in the "Service" folder and aims to handle all business logic of the application. Every REST API controller has its business logic class, *NameService*, where the **Name** is the controller's name.

REST API layer. REST API layer is located in the "Controller" folder and was created as ASP.NET Core Web API Controllers to define routings to be called to process the operations. To every Controller, the appropriate Service class is injected using dependency injection, which carries all business logic.

Presentation layer. A presentation layer is a client web page application named as **EvaluationClient** that communicates with the server using REST API.

3.9 Authentication & Authorization

This section describes the solution to the authentication and authorization problem in the application using the default ASP.NET Core Identity framework [57] with the Bearer access token technology [58].

The Bearer token is a string acting as the authentication of an API request, which is sent in the HTTP header as "Authorization" parameter. This string has no meaning to clients using it and no predetermined length. Requiring no cryptographic signing, Bearer tokens prove to be one of the simplest ways of making API requests. Due to the fact that a request includes a token as plaintext which once intercepted can be used by anyone, it must use HTTPS connection. The Bearer token offers an obvious advantage for both clients and servers as it does not require complex libraries to make requests and is easy to use.

The fact that there is nothing preventing other applications from using the Bearer token is a distinct disadvantage. Despite this common criticism, most providers still use the Bearer token anyway. Due to the fact that under normal circumstances, when application is properly protected, i.e. the access token is under its control, this is not an issue. However, technically the Bearer token is less secure.

3.10 EvaluationClient

This section deals with different evaluation rule types in the client application, the coherence and sequencing between different rule group types.

Figure 3.6 shows the initial state of the application. In this state the user can define root evaluation rules. There is a six-icon toolbar above the table

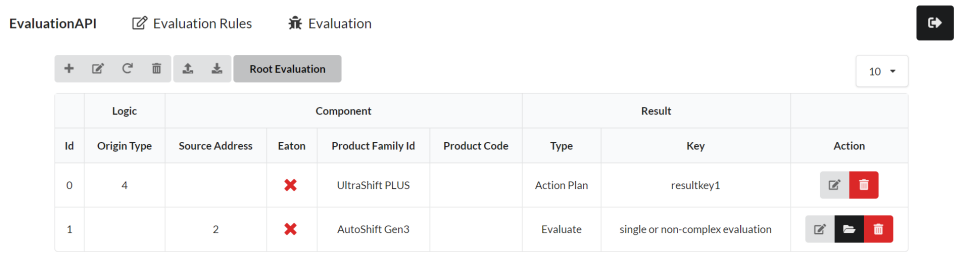

Figure 3.6: Root Evaluation Rule Group

of rules that enables the user to initialize new rule group, enter edit mode, refresh rules, delete particular rule group, import or export rules. Moreover, next to the toolbar goes overview of the current state of the application. Its use-case is vivid in figure 3.9, where the sequencing and sub-sequencing of the rule groups are shown.

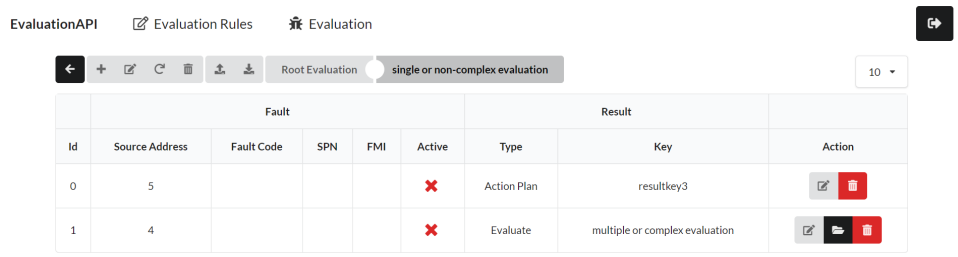

Figure 3.7: Single/Non-Complex Evaluation Rule Group

The priority of rules can be changed by using the "Edit" icon in the toolbar by dragging and dropping rules in the table. The "Action" column in the table contains "Edit" and "Delete" icons for every rule, but for rule with Result Type "Evaluate" "Open" icon is added. By clicking the latter the user is directed to the rule group with the same name as rule's Result Key, i.e. to the table with Single/Non-Complex Evaluation Rules shown in figure 3.7.

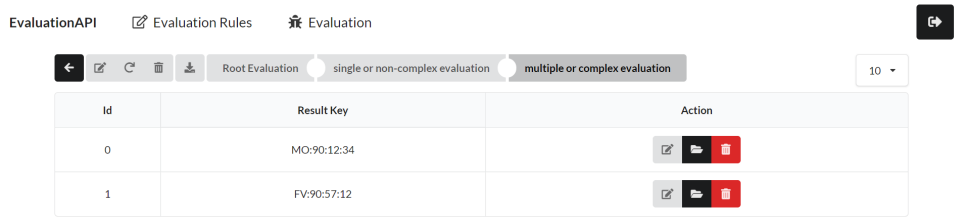

Figure 3.8: Multiple/Complex Evaluation Rule Group

If by clicking "Open" button, the application detects no rule in opened rule group it offers to initialize it in two possible ways: simply initialize an empty rule or import rule(s) from an OOXML file. After successful initialization the application directs the user to the newly created rule group. If evaluation sequence requires complex rules, the user can create a Complex Evaluation

Rule Group shown in figure 3.8. This type of rule group represents the set of complex rules where each rule can be opened using "Open" button for further definition process. Figure 3.9 shows how each complex rule is represented in the application.

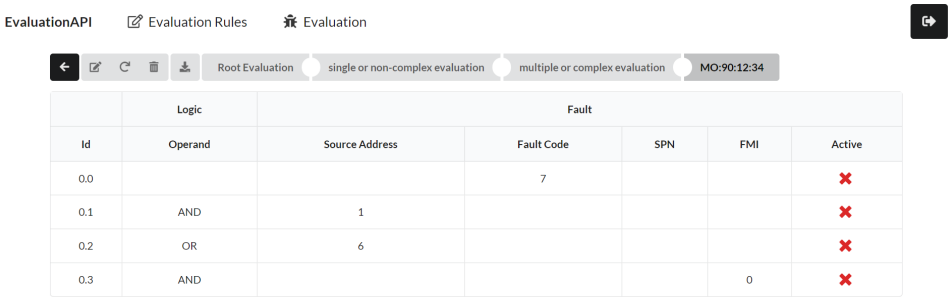

Figure 3.9: Complex Evaluation Rule Example

The complex rule is the set of non-complex rules that together define complex expression using Prefix and Suffix. The application automatically generates right Suffix so that the user just needs to choose appropriate Operand/Prefix.

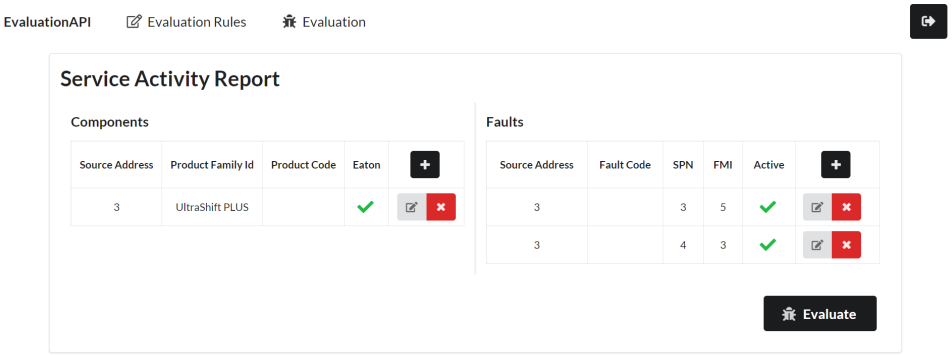

Figure 3.10: Service Activity Report Evaluation

After defining the evaluation rules and their sequence and interdependence the user is able to create and evaluate SAR request presented on the "Evaluation" page shown in figure 3.10, i.e. simulate vehicle components and faults to check the rules correctness.

Conclusion

The present thesis has set as its aim the analysis of the existing part of the application(IntelliConnect) for evaluation of vehicle components faults and the design of an improved version with UI. In order to do so, it was necessary to explore what drawbacks exist, what functional and non-functional requirements are, and define architecture and technologies to be used.

Having analyzed the existing solution several drawbacks have been detected:

- The evaluation rules are modifying in OOXML files. The database is changed using a simple console application that enables just to import evaluation rules from an OOXML file.
- The priority of evaluation rules is changing manually. In case that there is a lot of rules this process can take a while.
- There is no option to automatically validate the correctness of the evaluation rules.
- There is no UI.

The new implementation that has been created within the project makes it possible to eliminate the drawbacks mentioned above. The current process of the OOXML table manual transfer has been substituted for client application with a user interface that communicates with the server using public API. Evaluation rule definition has been simplified by adding a website tool for their modifications and automated correctness verification. Moreover, a dynamic view of fault codes has been developed. Now it is possible to simulate a group of fault codes and calculate a given severity. The old option to import rules in OOXML format has been saved and advanced. Additionally, the possibility to export evaluation rules in the same format has been added. Finally, simulation of fault codes to check the correctness of evaluation rules logic has been developed.

Even though there is still some possibility to improve the implemented solution, the results of the work have been done in the right direction to a sustainable and scalable system that the company Eaton can use to optimize and automate the process of modifying and evaluating rules.

From an architectural perspective, it is a system consisting of three layers: presentation layer, business logic and data layer. The layered architecture has been followed to achieve sustainability, scalability and testability of the application.

The application website page is designed to be used by technical staff. Intuitive and simple operation together with a clear interface ensures easy use of the system. The client application aims to enable technical staff to manage, evaluate, and validate rules, thus optimize and automate all manual processes. In terms of functionality, the client application meets the specified requirements and is completely usable.

The results obtained have led us to think about some possible future improvements. Realizing that the biggest shortcoming of this work is the absence of automated testing, which is beyond this project's scope, we still believe that its development would greatly contribute to the application's reliability and sustainability. Another possibility for further development is to upgrade faults evaluation outcomes. This can be achieved by improving the present results of the evaluation process, i.e. action plan key may be provided with a detailed action plan. One of the other future enhancements is adding the admin's role, who would manage user's roles and perform advanced operations on rule groups, including deleting all rule groups and migration of original rules to the new database.

Bibliography

[1] Steven Parissien. *The life of the automobile: the complete history of the motor car*. Macmillan, 2014.

[2] Life in Roztoky. `https://www.eaton.com/cz/en-gb/company/careers/life-at-eaton/life-in-roztoky.html`.

[3] Softing. `https://company.softing.com/`.

[4] Markus Steffelbauer. Security challenges in diagnostics 4.0.

[5] Wabco. `https://www.transics.com/`.

[6] TX-TRAILERPULSE. `https://www.transics.com/product/tx-trailerpulse/`.

[7] Eaton IntelliConnect™ - remote diagnostics. `https://www.eaton.com/Eaton/ProductsServices/Vehicle/intelliconnect/index.htm`.

[8] IntelliConnect. `https://www.eaton.com/Eaton/ProductsServices/Vehicle/tools/IntelliConnect/index.htm`.

[9] Christof Ebert and Capers Jones. Embedded software: Facts, figures, and future. *Computer*, 42(4):42–52, 2009.

[10] Karl Henrik Johansson, Martin Törngren, and Lars Nielsen. Vehicle applications of controller area network. In *Handbook of networked and embedded control systems*, pages 741–765. Springer, 2005.

[11] N. Navet. Controller area network [automotive applications]. *IEEE Potentials*, 17(4):12–14, 1998.

[12] Steve Corrigan HPL. Introduction to the controller area network (can). *Application Report SLOA101*, pages 1–17, 2002.

[13] Keith Pazul. Controller area network (can) basics. *Microchip Technology Inc*, 1, 1999.

[14] Alison Quine, January 2008. Carrier Sense Multiple Access Collision Detect (CSMA/CD) Explained. `https://www.itprc.com/carrier-sense-multiple-access-collision-detect-csmacd-explained/`.

[15] H. Chen and J. Tian. Research on the controller area network. In *2009 International Conference on Networking and Digital Society*, volume 2, pages 251–254, 2009.

[16] Peter Bagschik. An introduction to can. *Braunschweig, Alemanha. Checked in: http://www. ime-actia. com/can_intro. htm. Last access*, 11:28, 2001.

[17] `www.sae.org`, homepage of the organization SAE International.

[18] Konrad Etschberger, Roman Hofmann, Joachim Stolberg, Christian Schlegel, and Stefan Weiher. *Controller area network: basics, protocols, chips and applications*. IXXAT Automation, 2001.

[19] `www.can-cia.de`, homepage of the organization CAN in Automation (CiA).

[20] `www.rockwellautomation.com`, homepage of the organization Rockwell Automation.

[21] `www.kvaser.com`, homepage of the organization Kvaser.

[22] Controller area network. can history. `https://technodocbox.com/Computer_Networking/70891460-Controller-area-network-can-history.html`. (Accessed on 11/28/2020).

[23] Jaykrishna Joshi, Dattatray Bade, and Sanjay Hundiwale. Car automation using controller area network.

[24] Uwe Kiencke, Siegfried Dais, and Martin Litschel. Automotive serial controller area network. *SAE transactions*, pages 823–828, 1986.

[25] CAN Specification. Bosch. *Robert Bosch GmbH, Postfach*, 50, 1991.

[26] Joaquim Ferreira and José Fonseca. Controller area network. *Industrial Electronics Handbook, Industrial Communication Systems*, 2011.

[27] Dan Noonen, Stuart Siegel, and Pat Maloney. Devicenettm application protocol. 1994.

[28] Olaf Pfeiffer, Andrew Ayre, and Christian Keydel. *Embedded networking with CAN and CANopen*. Copperhill Media, 2008.

[29] Yelizaveta Burakova, Bill Hass, Leif Millar, and André Weimerskirch. Truck hacking: An experimental analysis of the SAE j1939 standard. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, August 2016. USENIX Association.

[30] Wilfried Voss. SAE J1939, Serial Control and Communications Vehicle Network. esd electronics, Inc. `http://www.esd-electronics-usa.com/online-seminars.html`.

[31] PGNs and SPNs look up table. https://www.isobus.net/isobus/pGNAndSPN.

[32] Souce Addresses look up table. https://www.isobus.net/isobus/sourceAddress.

[33] Yang Jiansen, Guo Konghui, Ding Haitao, Zhang Jianwei, and Xiang Bin. The application of sae j1939 protocol in automobile smart and integrated control system. In *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, volume 3, pages 412–415, 2010.

[34] Xuan Shao, Xingwu Kang, Xuping Wang, and Xiaojing Yuan. Design of special vehicle condition monitoring system based on j1939. In *Journal of Physics: Conference Series*, volume 1549, page 032092. IOP Publishing, 2020.

[35] Suspect Parameter Numbers (SPN) on J1939 data link. https://lnx.numeralkod.com/wordpress/docs/errors-index/suspect-parameter-numbers-spn/.

[36] Failure Mode Identifier (FMI) Codes on J1939 data link. https://lnx.numeralkod.com/wordpress/docs/errors-index/failure-mode-identifier-fmi-codes-on-j1939-data-link/.

[37] Eaton Transmission Families. https://www.eaton.com/Eaton/ProductsServices/Vehicle/Transmissions/index.htm.

[38] Eaton Endurant automated transmission. https://www.eatoncummins.com/us/en-us/catalog/transmissions/endurant.html.

[39] Eaton Fuller Advantage automated manual transmission. https://www.eatoncummins.com/us/en-us/catalog/transmissions/fuller-advantage-automated.html.

[40] Eaton UltraShift PLUS automated manual transmission. https://www.eatoncummins.com/us/en-us/catalog/transmissions/ultrashift-plus-mhp.html.

[41] Eaton Procision https://www.eaton.com/Eaton/ProductsServices/Vehicle/Expertise/expert-articles/procision-delivers/index.htm.

[42] Eaton AutoShift https://www.eaton.com/Eaton/ProductsServices/Vehicle/Transmissions/heavy-duty-automated/autoshift/index.htm.

[43] IntelliConnect: IoT innovation at every turn. https://www.eaton.com/us/en-us/company/news-insights/internet-of-things/intelliconnect.html.

[44] Eaton IntelliConnect portal site. https://eatonintelliconnect.com/.

[45] Peer-to-Peer and Client-Queue-Client Architecture http://www.exforsys.com/tutorials/client-server/peer-to-peer-and-client-queue-client-architecture.html.

[46] ASP.NET Core https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core.

[47] ASP.NET https://dotnet.microsoft.com/apps/aspnet.

[48] ReactJs https://reactjs.org/.

[49] VueJs https://vuejs.org/.

[50] PostgreSQL https://www.postgresql.org/.

[51] Microsoft SQL Server https://www.microsoft.com/en-us/sql-server/sql-server-2019.

[52] Semantic-UI-React https://react.semantic-ui.com/.

[53] Visual Studio Enterprise Edition https://visualstudio.microsoft.com/vs/enterprise/.

[54] Microsoft SQL Server Management Studio https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15.

[55] Intellij Idea Webstorm https://www.jetbrains.com/webstorm/.

[56] Azure DevOps https://azure.microsoft.com/en-us/services/devops/.

[57] ASP.NET Core Identity https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio.

[58] Bearer Token Authentication in ASP.NET Core https://devblogs.microsoft.com/aspnet/bearer-token-authentication-in-asp-net-core/.

Appendix A

Acronyms

API. Application Programming Interface

CAN. Control Area Network

CSMA/CD. Carrier Sense Multiple Access with Collision Detection

ECU. Electronic Control Unit

EEIC. Eaton European Innovation Center in Prague

EKMS. Eaton Knowledge Management System

HTML. HyperText Markup Language

HTTP. Hypertext Transfer Protocol

HTTPS. Hypertext Transfer Protocol Secure

ISO. International Standard Organisation

JSON. JavaScript Object Notation

LAN. Local Area Network

LINQ. Language Integrated Query

MSSQL. Microsoft SQL Server

OOXML. Office Open XML

ORM. Object–Relational Mapping

OSI. Open Systems Interconnection

P2P. Peer-to-Peer

REST. Representational State Transfer

SAE. Society of Automotive Engineers

SAR. Service Activity Report

SQL. Structured Query Language

UI. User Interface

VCS. Version Control System

WPF. Windows Presentation Foundation

XHTML. Extensible HyperText Markup Language

XML. Extensible Markup Language

Appendix B

Contents of attached CD

The contents of CD is organized into the following files and folders

- **evaluationapi.zip** - ASP.NET Core server-side of the application.
- **evaluationclient.zip** - ReactJs client-side of the application.
- **bachelor_thesis.pdf** - bachelor thesis in pdf format.