Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Matej Mnoucek

Data-oriented Multi-agent Assessment System for Real-time Driving Simulators

Master's thesis in Informatics
Supervisor: Odd Erik Gundersen
July 2020

NTNU
Norwegian University of Science and Technology

Matej Mnoucek

Data-oriented multi-agent assessment system for real-time driving simulators

Master thesis, Spring 2020

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

Interactive driving simulators are slowly becoming a technique used for education and training of future drivers. However, these systems need to be operated by human driving teachers who provide the actual educational value and feedback to students. This limitation lowers the degree of autonomous operation of driving simulators and puts additional requirements on their system design. In addition, human resources are usually expensive. The situation is further complicated by the fact that interactive simulations usually require a lot of computational power and operate in real-time, therefore, any additional supplementary systems might need to run in a resource constrained environment.

This document proposes a novel data-oriented multi-agent assessment system designed for real-time driving simulators capable of providing feedback about driving skills while also teaching students about the traffic domain. The system is designed to operate with high efficiency and in real-time while providing the output necessary to guide and educate a student driver. The correctness of the implementation is validated by two experiments designed to emulate real world traffic scenarios in order to obtain reliable system verification. Hence, the main contributions of this work consist of the novel data-oriented assessment system which contains 18 intelligent assessment agents that can be further enhanced, replaced or new ones can be added as well.

Preface

The following document contains a master thesis which was produced as the final work required to acquire Master of Science degree in Informatics at the Norwegian University of Science and Technology. The project was done in collaboration with Way As company, which provided us with the necessary technical resources, access to their systems and a driving simulator for testing purposes. Therefore, special thanks goes to all Way AS employees for their endless helpfulness, effort and patience. I would also like to thank Odd Erik Gundersen for his great supervision, knowledge contributions and valuable feedback provided. In addition, I am really grateful to Martin Kristoffer Hoel Sandberg for his enormous enthusiasm and support during our long evening discussions.

The thesis explores the field of Data-oriented design and real-time interactive simulations combined with ontology-based multi-agent systems applied to the domain of driving simulators. The aim is to provide a solution which bridges the gap between a complex artificial intelligence-driven system and real-time operation constrained system.

Matej Mnoucek Trondheim, July 9, 2020

Contents

Acronyms

2D two-dimensional 33

ACL Agent Communication Language 14

ADAS Advanced Driver Assistance Systems 36, 39

AR Augmented reality 21

CPU Central Processing Unit 10

DL Description Logic 19, 38, 41

DoD Data-oriented design 10–12, 34, 42

DRAM Dynamic Random Access Memory ix, 10, 11

DSR Design Science Research ix, 3, 5

ECS Entity Component System ix, x, 12, 13, 29–31, 33, 34, 40, 42, 54–56

ECV Eventually Consistent Vector x, 55, 57

FOL First Order Logic 19

ITS Intelligent Tutoring Systems 35, 101

M2M Model-to-Model 17

M2T Model-to-Text 17

MDA Model-Driven Architecture 17

MDD Model-Driven Development 17

MLN Markov Logic Network 37

OoD Object-oriented design 34

OoP Object-oriented programming 10, 19, 33

OWL Ontology Web Language 16, 17, 19, 31, 34, 36, 38, 39, 41

PBIL Population-Based Incremental Learning 35

RDF Resource Description Framework 16, 31, 36

RPM Revolutions Per Minute 70

SA Situation Awareness 14

SC Situational Calculus 17

SG Serious Games 35

SLR Structured Literature Review xiii, 23, 24, 26, 27

SWRL Semantic Web Rule Language 31, 34, 36, 38

TR Temporal Reasoning 17

UAV Unmanned Aerial Vehicle 32

VANET Vehicular Ad-hoc NETwork 31

VR Virtual reality 21

XML Extensible Markup Language 16, 30

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an insight into the background and motivation behind this kind of work. Besides, it contains specification of the main goal, defines research questions to answer, provides description of the employed research methods, describes main contributions and concludes with a description of the document structure.

1.1 Background and Motivation

Interactive driving simulators are slowly becoming a technique used for education and training of future drivers. However, these systems need to be operated by human driving teachers who provide the actual educational value and feedback to students. This limitation lowers the degree of autonomous operation of driving simulators and puts additional requirements on their system design. In addition, human resources are usually expensive. The situation is further complicated by the fact that interactive simulations usually require a lot of computational power and operate in real-time, therefore, any additional supplementary systems might need to run in a resource constrained environment.

For these reasons, the existence of an efficient system capable of real-time autonomous driving skills assessment would push interactive driving simulators to an entirely new level. It would allow separation of human teachers from the simulator experience, therefore, making simulator-based learning independent, cheaper, more consistent and more efficient. In addition, the research is conducted in collaboration with Way As driving school company, which is deeply interested in exploring the possibilities of the autonomous assessment as well.

1.2 Research Context

The research was conducted at the Department of Computer Science at the Norwegian University of Science and Technology and also in collaboration with Way As, a driving school based in Trondheim. On top of normal driving education, Way offers driving lessons in their custom designed and developed driving simulator. The simulator consists of a physical car mounted on a moving platform surrounded by 360° simulated environment projection. Moreover, the car also houses several kinds of force feedback systems to ensure maximal realism and immersion. The driving simulator software and hardware were freely available for the whole period of collaboration enabling close cooperation and extensive testing and verification of the artifacts produced as results of this research.

Figure 1.1: Driving simulator developed by Way AS.

During the time of the research work described in this thesis, there were in fact three parallel research efforts going on at Way As. The first research effort was focused on developing a proof of concept ad-hoc assessment system with minimal effort. The plan was to embed the system in the current simulator software. The second effort was concerned about building standalone, efficient and data-oriented multi-agent assessment system which uses ontologies as simulated environment abstraction. In addition, there was a focus on supporting a wide variety of temporal and logic reasoning, developing a proper simulator interface and creating a

good and scalable foundation for future development. This part of the research was conducted by us and is described in this thesis. The last effort concentrated on developing a virtual driving instructor capable of presenting the results produced by the assessment systems. The goal of the instructor system was to deliver an appropriate feedback to driving students at appropriate times. This part of research was conducted by Martin Kristoffer Hoel Sandberg as is described in his master thesis [7].

1.3 Goals and Research Questions

The global aim of this thesis is to explore the possibilities of creating a real-time data-oriented multi-agent assessment system. Therefore, this gives a rise to the main research driving hypothesis:

Hypothesis *Data-oriented multi-agent system for driving skills assessment in a simulated environment can be designed to run in real-time while supporting all functionality required for complete and timely driving skills evaluation.*

The hypothesis enables us to set the overall project goal and pose several related research questions:

Goal *Design and develop a data-oriented multi-agent assessment system which is capable of operating in real-time while providing the targeted driving skills evaluation.*

Research question 1 *Which data-oriented design principles can be utilized for the design of the assessment system?*

Research question 2 *How can the system reason about traffic situations?*

Research question 3 *How can the system utilize concurrency on multi-core systems?*

Research question 4 *How can the system interface with a driving simulator in order to extract the necessary data?*

Research question 5 *Is it possible for such system to deliver driving skills assessment in real-time?*

1.4 Research Method

The research documented in this thesis combines two different research strategies, namely, *Design Science Research (DSR)* methodology as summarized and

described by Vaishnavi and Kuechler [8] and experiments [9]. The former methodology encapsulates the process of assessment system design and implementation whereas the latter deals with its functional verification and evaluation. The fundamental goal of the second employed strategy is to provide measurable and detailed insight into how the designed system performs in real-life scenarios (see chapter 4). Both methodologies combined are expected to deliver concise and complete answers to the initially stated research questions. The research was conducted in five major phases which mirror the standard process of Design Science Research:

1. **Awareness of Problem:** The problem definition was supplied as a master thesis proposal, therefore, this step was mostly neglected in the actual research process. However, the main hypothesis, goal and research questions were defined in this phase (section 1.3).

2. **Suggestion:** During the suggestion phase a tentative solution proposal was made based on a configuration of several new and existing approaches and techniques. In addition, traffic situation ontology/model was produced as the first artifact (subsection 3.1.1).

3. **Development:** The development phase was concerned with the actual implementation of the assesment system design proposed in the previous phase. Hence, the first result was a system instantiation forming the primary output artifact (section 3.2). In addition to the assessment system, 18 demonstrative intelligent agents were developed in order to prove and verify the capabilities of the system. The agents themselves constitute a second instantiation produced during this phase (section 3.4).

4. **Evaluation:** The evaluation phase is when the second methodology i.e. experiments gets involved. The experiments were carried out in order to derive and verify answers to the initially posed research questions (chapter 4). The evaluation itself was carried out through interaction with domain experts i.e. driving instructors in a form of interviews and questionnaires. Two driving lesson scenarios and the produced real-time assessment were inspected, tested and analyzed by the domain experts. Both quantitative and qualitative evaluation methods were utilized to acquire the final evaluation results.

5. **Conclusion:** This phase is the finale of the whole research effort. The results were consolidated, discussed and compared to expected outcomes (chapter 5). Deviations from expected results and various findings made during the research process were discussed and addressed.

1.5 Contributions

According to *Design Science Research Knowledge Contribution Framework* proposed by Gregor and Hevner [1], this research falls somewhere on the edge between *Exaptation* and *Invention* quadrants as it utilizes innovative adaptation of known knowledge/solutions as well as inventing new knowledge/solutions for new problems.

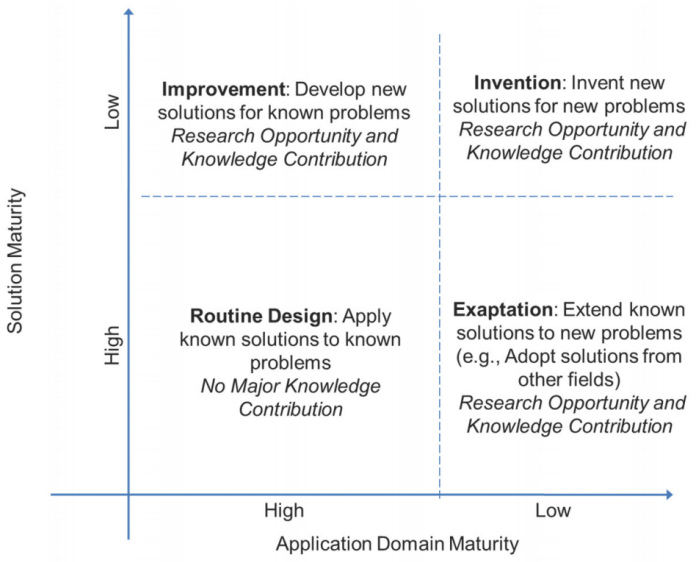


Figure 1.2: DSR Knowledge Contribution Framework [1]

The presented research includes several individual contributions. For a better readability and clarity they were organized in the list below:

1. **Temporal ontology suitable for real-time traffic situation knowledge representation:** As the assessment system requires to sense the virtual environment and is constrained to running in real-time, there was a need to create a suitable knowledge representation which satisfies both requirements. In addition, the ontology is required to store its past states as well as to allow reasoning over time. For that purpose, a new high level ontology was developed which during compilation gets transformed into an efficient low level data-oriented representation capable of usage in a real-time system (section 3.1).

2. **Pattern query engine capable of temporal queries on the ontology in real-time:** The ontology mentioned in the previous contribution also needs to allow efficient data queries. To address this problem, custom pattern query engine inspired by Description logic was designed and developed. The query engine allows agents to interact with the ontology and efficiently query the knowledge of interest (subsection 3.1.5).

3. **Road network marking tool for simulated environments designed in Unity development platform.** The simulated worlds, in which driving education takes place, usually host complex road networks. Road networks are one of the basic elements that the developed assessment system needs to obtain information about. Hence, as a part of the implementation, universal spline-based lane marking tool for easy road network annotation was developed (subsection 3.3.2).

4. **Wait-free shared memory interface facilitating assessment system communication with the simulated environment.** The nature of the solution, requires real-time communication between the newly developed assessment system and existing simulated environments i.e. there is a need for efficient data exchange. For this reason, wait-free shared memory storage solution for the real-time ontology was developed and utilized (subsection 3.3.4).

5. **Scalable data-oriented multi-agent assessment system design.** Before the assessment system was built, a system design proposal was created. The proposal is domain and implementation independent, universal and complete. Hence, it can be utilized in other domains too (section 3.2).

6. **Assessment system instantiation including 18 intelligent agents for basic traffic situation assessment.** The final contribution is the provided system and included agents implementation on its own. It is capable of basic traffic situation assessment and real-time Text-To-Speech feedback to a driver (chapter 3 and section 3.4).

1.6 Thesis Structure

This document is structured as follows:

Chapter 2 – Background Theory and Motivation: presents theoretical knowledge relevant to the aim of this work and also describes and provides results of the conducted literature review. Finally, the chapter discusses additional motivation for this work.

Chapter 3 – System Design and Implementation: provides details about the assessment system design and implementation process. Furthermore, the chapter justifies the design decisions made and links the work to the relevant previous research.

Chapter 4 – Experiments and Results: describes the experiments conducted with domain experts and presents the results of them in a form of quantitative and qualitative data.

Chapter 5 – Evaluation and Conclusion: discusses the achieved results and confronts them with the initially posed hypothesis, goal and research questions. Furthermore, limitations of this work and potential future work proposals are also mentioned.

Chapter 6 – Publications: lists publications produced as a part of this research effort.

Chapter 2

Background Theory and Motivation

In this chapter, we introduce the relevant background theory and present *Structured Literature Review* which was conducted to map out the existing research. Moreover, the chapter also addresses motivation behind this kind of work.

2.1 Background Theory

The goal of the following section is to present domains of the most relevant knowledge needed to frame and solve the problem of designing real-time multi-agent assessment system in a data-oriented manner.

2.1.1 Memory and Data Locality

At first, in order to understand Data-oriented design, it is necessary to know a bit about modern computers architecture, especially about *data locality* and *memory hierarchies* [10].

There exist two principles that have been employed to make memory access more efficient. The first assumption is called *temporal locality* and claims that if a memory cell is accessed at some point in time, there is a high probability that it will be accessed again soon. The second principle called *spatial locality* concerns the location of data in memory. The assumption behind it claims that memory cells neighbouring a cell which was just accessed are very likely to be accessed as well.

Nextly, a modern computer contains several levels of memory. The fastest memory (registers) is located closest to CPU but also is the smallest of them all. The main and largest memory is DRAM which usually has a great capacity but takes a long time to access. In between there are up to several levels of cache memories usually termed L1-L3 based on their location (L1 is the closest to CPU).

The memory hierarchy and locality principles guide the design of cache hierarchy and their internal workings. Everytime a program tries to access DRAM memory location the computer actually loads a whole piece of memory called *cache line* (or cache block) and store it in the corresponding cache. Then, when the program execution continues the loaded memory cell and also the neighbouring ones are already available in the fast cache memory so its desirable to make use of them. In case they really are available, the situation is called *cache hit* and the program can immediately continue. Otherwise, *cache miss* occurs. The program then has to wait until the needed data is available i.e loaded into cache.

The awareness of these principles and the resulting efficient cache utilization is one of the core foundations of DoD.

2.1.2 Data-oriented Design

Data-oriented design is a practice of developing software in which software is seen as a transformation of data from one form to another [11]. It also builds on the fact that the transformations are not performed in vacuum but are processed by hardware of some sort. The main concerns of this paradigm are building high performance software, capable of real-time and parallel execution which is easy to test and modify. Fundamentally, Data-oriented design (DoD) is guided by two essential principles.

The first principle claims that data is not the actual problem domain. In other words, DoD does not embed the problem domain into the code as other abstraction heavy paradigms commonly do. For example Object-oriented programming (OoP) pretends that the computer and its data does not exist and abstracts away from it. This approach often shadows the target platform characteristic and, therefore, hinders software performance. Another frequent problem is that OoP paradigm leads to piling up of unrelated data in classes and creates strong coupling between both data and behavior which is hard to unwind.

The second principle promotes that data is more than just a structure and all of its aspects should be considered. The important aspects to focus on contain quantity of data, access frequency, probability of access and similar statistics.

The values that emerge from these considerations help to guide efficient data layout design and ensure good cache memory utilisation.

There is another phenomenon which DoD attempts to address. According to Patterson et al. [2] the gap between processor and DRAM speed is increasing by 50% each year which is one of the primary obstacles to improved computer system performance. This problem can be partially mitigated by introducing deeper cache memory hierarchies, hence, efficient cache memory management keeps growing in importance. Figure 2.1 illustrates the problem.

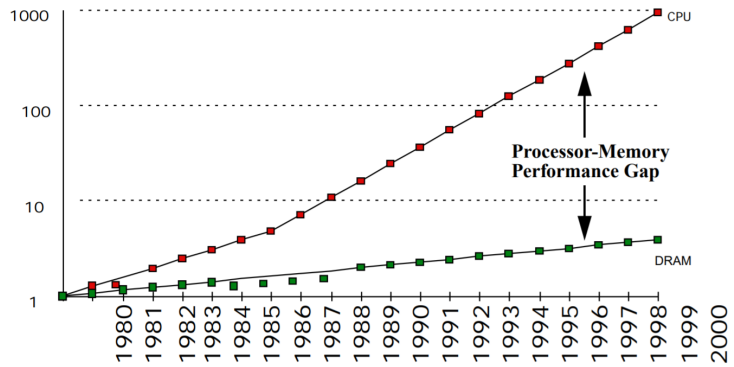


Figure 2.1: The continuously increasing gap between processor and DRAM performance [2].

One might think that a good portion of the issues with efficient data and memory handling should be addressed on the compiler level. However, according to Proebsting and Scott [12], the improvements and advances in compiler technology double typical software performance only roughly every 18 years. Hence, compilers are not able to compensate for the processor-memory performance gap.

There are several common techniques used in DoD which project the paradigm goals into actual good design practices [13, 11]:

DBMS-like memory layout – exploit of the advantages of data layout commonly used by DBMS i.e. two dimensional tables. This approach also concerns the creation of the layout from actual data and employs techniques such as normalisation. The layout provides a simple data structure which is cache efficient and can be easily iterated over.

Linear and continuous data structures – the use of simple and linear data structures such as vectors or arrays because of their cache efficiency and the ease of work parallelization.

Data packing and sorting – the assumption of sorting data by various statistics e.g. how often or in which order it is commonly used.

Hot/cold splitting – separation of frequently accessed data from the data that is rarely used.

Existential processing – avoiding checking/processing of data which does not need an update or any other handling.

Components-based architecture – separation of large "objects" or entities into isolated problem domains i.e. composition over inheritance.

In addition, DoD in general allows better separation between data and operations performed on them which improves software modularity and allows easier code refactoring. There already exist well-established architectural and design patterns built with data-oriented principles in mind as can be seen in the following chapter.

2.1.3 Entity Component System

One of the most common architectural patterns frequently used in Data-oriented software design is Entity Component System (ECS). ECS enforces composition over inheritance principle i.e. decomposition of "objects" into separate and independent components [14].

As the name suggests, ECS is build around three fundamental concepts: Components, Entities and Systems. Components are sole containers for data that represent some information. These could be for example a position in 3D world, speed velocity or any other elemental piece of data. Entities are analogous to objects in Object-oriented programming. They represent groupings of components and, therefore, form higher level concepts. However, they are often very lightweight and commonly represented as just a unique identifier which serves as an index into component collections. Systems are the modules where logic resides. They provide access to entities and perform operations on them by manipulating their component data. Figure 2.2 illustrates the commonly employed memory layout for entities and their components i.e. two dimensional column-wise array

containing individual components.

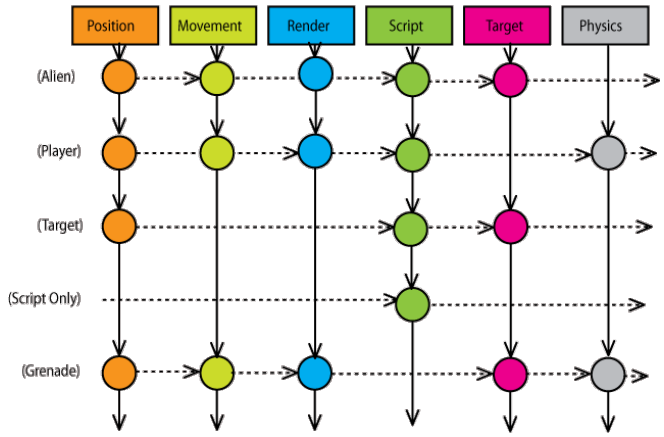


Figure 2.2: Example ECS component memory layout of a computer game. The names in parentheses represent individual entities, the colored dots are components and the rectangles stand for individual systems [3].

2.1.4 Intelligent Agents and Multi-agent Systems

Wooldridge [15] defines an Intelligent agent as a computer system that is situated in some environment, and is capable of autonomous actions in this environment in order to meet its objectives. The autonomousness stands for agent's constant awareness of its objectives and his everlasting effort to complete them in the most efficient and effective way [16]. The agent's environment is often continuous, dynamic and non-deterministic which further hinders agent's efforts [17]. The basic agent data flow starts with perception of its environment and results in the production of the desired data output. In order to produce the output, agents often need to maintain some internal state which captures the history of their actions and knowledge about their environment. Potentially, they also store some initial knowledge about the domain of the problem being solved.

If we group several agents together, the system at hand becomes multi-agent. The presence of more than one agent implies several new facts. Individual agents start to influence each other. Also, even though the agents remain independent, they might depend on some data produced by other agents. This fact usually implies a need for some inter-agent communication via Agent Communication

Language (ACL) as there is a data sharing demand.

The way how agents coexist in terms of problem solving also matters. Agents in multi-agent environment might either collaborate or compete with each other based on what their goals are. The distinction can be seen as an effort to maximize shared utility of all agents (*collaborative*) or each agent's individual utility (*competitive*) [18]. This work is mainly concerned about collaborative multi-agent systems.

2.1.5 Situation Awareness

The correct assessment of a particular traffic situation requires its perception and deep comprehension i.e. high degree of situation awareness. Endsley [19] defines a formal theoretical model of situation awareness in relation to human decision making. The model is shown in Figure 2.3. It defines decision making as a three step process. The first step deals with obtaining situation awareness, then the actual decision takes place and finally the performance of taken actions is evaluated. This thesis is essentially concerned only about the first step i.e. to create a system which is able to obtain situation awareness and reason about it. The situation awareness step is further divided into three sub steps so called levels:

1. **Perception of elements in current situation:** The first step in achieving SA is to perceive the status, attributes, and dynamics of relevant elements in an environment within a volume of time and space.

2. **Comprehension of current situation:** The second step goes beyond simply being aware of the elements that are present. It focuses on understanding the significance of these elements in a broader scale, relates them together to discover patterns and builds comprehension of their meaning.

3. **Projection of future status:** The third step encapsulates the projection of the status of elements in the near future.

2.1.6 Ontologies and Knowledge Graphs

The term *Ontology* was initially adopted from philosophical sciences [17]. Ontology explains the nature and properties of individual concepts and captures the relationships between them [16]. The main use case of ontology is to represent and store domain knowledge in a consistent manner. Hence, *ontological commitment* is enforced i.e. an agreement about what the concepts and relationships between them mean in the real domain. Moreover, there often are additional

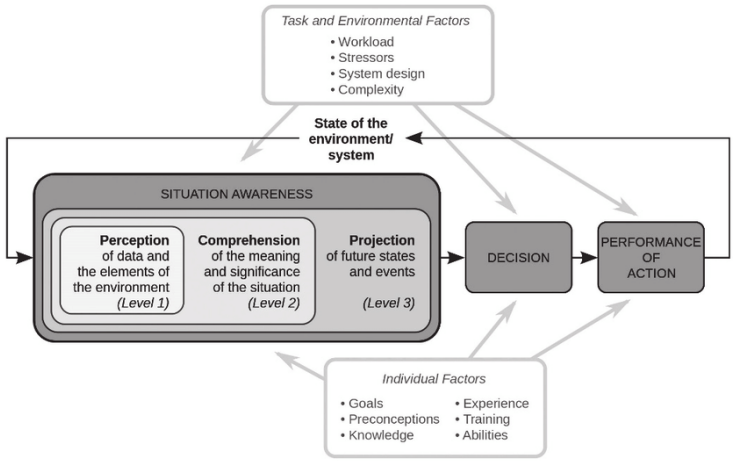


Figure 2.3: Model of situation awareness in dynamic decision making.

agreements ensuring coherent and consistent use of the shared vocabulary (usually axioms and definitions). Moreover, these sets of objects and relationships are often called *the universe of discourse* [16].

Ontologies often involve a processes called generalization or specialisation. Specialisation stands for making an ontology more specialized i.e. narrow down its scope and increase the amount of detailed information it contains about the new more narrow domain. Generalisation is the inverse of specialisation.

Typically, an ontology forms a hierarchical structure which describes domain concepts, their properties and captures relationships between them [16]. The main reason for the hierarchical organization is the similarity with how human mind organizes things and it also allows easy modelling of generalization and specialisation relationships. In fact, the goal of the hierarchy is to capture how concepts are organized into predefined categories [17]. Categories can be subcategories of other categories. This fact allows subcategories to inherit properties or relationships of their parent categories which results into a formation of *taxonomy*.

Matheus et al. [4] proposed a definition of core ontology suitable for representing various scenarios of situation awareness (see subsection 2.1.5). The paper also proves the expressiveness of the proposed ontology and demonstrate its extensibil-

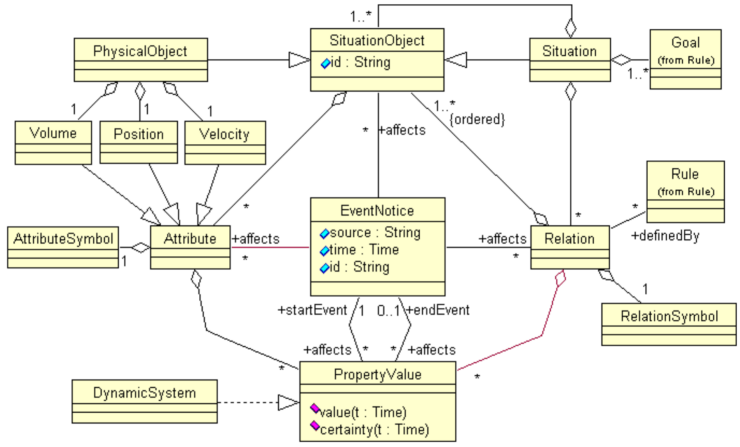


Figure 2.4: Core situation awareness ontology as proposed by Matheus et al. [4]

ity to domain-specific situations by readily extending its core language. Another important feature is the possibility to represent values of attributes and relations as changing over time and space. The core ontology is shown in Figure 2.4

Similarly to ontology definition, we can define *Knowledge graphs* which share many properties with ontologies. The main difference between knowledge graphs and ontologies is their structural aspect. As the name suggests, knowledge graphs can be represented as various graph structures [20]. In contrast, ontologies mostly form sole tree hierarchies.

The use of ontology or knowledge graphs in combination with multi-agent (subsection 2.1.4) environments allows the creation of a shared and consistent knowledge base. The knowledge base can then be utilized for mutual communication between agents.

One of the most widely-used languages for modelling ontologies is the Web Ontology Language (OWL). It relies on RDF/XML format which can be easily understood by computers.

2.1.7 Model-Driven Development and Transformations

Model, modelling and Model-Driven Architecture (MDA) are the core building blocks of Model-Driven Development (MDD). MDD itself is defined as a set of development practices and approaches where models are used to reason about a problem domain and design a solution in the solution domain [21]. Abstract models serve as a representation of the domain of interest and allow developers to easily convert the representation from one type into another (Model-to-Model or M2M transformation) or alternatively to the final source code representation (Model-to-Text, M2T transformation or code generation) [22]. These steps are usually referred to as *model transformations* which form a cornerstone of MDA. Nowadays, the transformations are usually performed automatically based on predefined transformation rules. These rules may be implicit to the tools being used, or may have been explicitly defined based on domain specific knowledge [21].

Another common term used in MDD is the term *metamodel*. Brown et al. [21] describe metamodels as models intrinsic to a modeling approach which basically form a framework for the modelling process. An example of commonly used metamodels is UML or an ontology defined in OWL [23]. Placing ontology as a source model in model transformations is a particularly interesting approach for this work as there is a need to derive an efficient implementation from high-level abstract ontology model. Furthermore, the implementation is required in several programming languages.

2.1.8 Temporal Representation and Reasoning

The representation of time and reasoning about it poses a problem which spans a wide range of disciplines including artificial intelligence. Vila [24] defines *Temporal Reasoning (TR)* as a formalization of the notion of time which provides means to represent and reason about the temporal aspect of knowledge. TR usually considers two main types of temporal primitives: *instants* or *points* and *periods* or *intervals* and also deals with various relations between them. Moreover, TR tries to answer various questions which constitute the structure of time such as: *Is time discrete or dense* or *Is time bounded or unbounded?* [25].

One of the earliest approaches to formalization of TR was presented by McCarthy and Hayes [26] in 1969. The representation was called *Situational Calculus*. In SC the world is represented as a set of *states*. State is a description of the represented world at a given instant of time. The world persist in a state until an action is performed. *Actions* were modelled as simple state transitions. In order to manage execution of actions, *fluents* were introduced. The domain of interest is described by *propositional fluents* which represent the properties that hold in

particular situations. *Propositional fluents* map to either *true* or *false*. There also exist *situational fluents* which represent changes by mapping situations to other situations. Time is handled implicitly through the notion of situations.

Another approach was presented by Allen [27]. The theory proposes a formalism based on the notion of *interval*. Interval is defined as the only temporal primitive while completely excluding points or instants. In addition, 13 mutually exclusive relations between intervals were proposed, namely: *before, after, meets, met-by, overlaps, overlapped-by, starts, starts-by, finishes, finished-by, during, contains* and *equals*. Figure 2.5 illustrates all the relations with a reference to global timeline. In this framework time is modelled as a linear, continuous, infinite and symmetric to past and future.

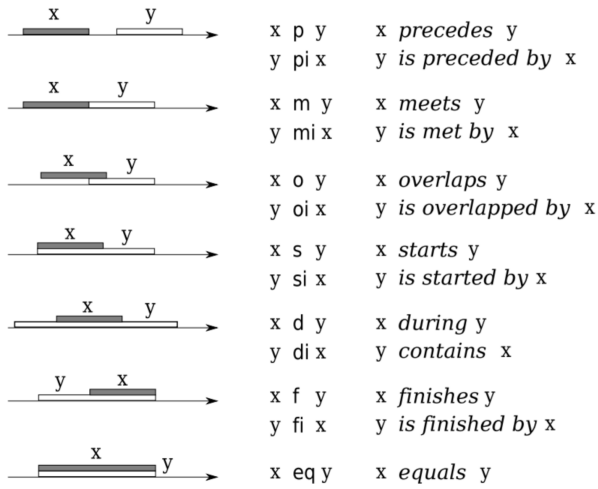


Figure 2.5: Interval relations in Allen's Interval Algebra [5]

The last influential approach was developed by Kowalski and Sergot [28]. Their theory called *Event Calculus* is based on the management of database updates where a simple time ontology imposes an extra level of semantics. The core ontological elements are *events* and *relations*. An event is considered to be anything that creates or deletes relations. A relation is a standard relation bounded by time in existence, therefore, implicitly defining a *time period*. For reasoning purposes, the theory uses a logic based on Horn clauses extended with negation-by-failure.

2.1.9 Description Logic and Reasoning

According to Krötzsch et al. [29] *Description logic* (DL) is a family of knowledge representation languages which are widely used in ontological modelling.

Description logic similarly to ontologies concerns reasoning about categories i.e. it makes easy to create definitions of categories and their properties [17]. The basic building blocks of DL are: *concepts*, *roles* and *individuals*. Concepts are similar to classes in OoP, roles represent relations between individuals and individuals are analogous to instances in OoP. In DL the paticular state of the world is defined by so-called *axioms* which can be divided into three categories: assertional ($ABox$) axioms, terminological ($TBox$) axioms and relational ($RBox$) axioms [29]. DL is closely related to *First order logic* (FOL) and can in fact be converted into FOL. Basically, concepts represent unary predicates, roles binary predicates and individuals akin to constants.

ABox axioms hold knowledge about individuals. They can be further divided into *concept assertions* and *role assertions*. Concept assertions relate individuals to a particular concept i.e. connects an instance to its class. Role assertions describe relations between individuals. TBox axioms define relationships between concepts and also can be categorized further. *Concept equivalence* and *concept inclusion* expresses that two concepts are equivalent or subsumed by the parent one respectively. RBox axioms suspectedly concern relations between individuals. Similarly to TBox axioms, they also allow *role equivalence* and *role subsumption*. In addition, *role composition* allows creation of new roles through combination of existing ones and *disjoint roles* forbid an existence of conflicting roles. To provide an analogy, TBox axioms are like a database schema and ABox axioms like the data it contains [30].

In DL, the execution of inference procedures is called *reasoning*. The basic inference tasks are *subsumption* i.e. checking if one category is a subset of another and *classification* i.e. checking whether object belongs to a certain category [17].

Additionally, Description logic is one of the core pillars of OWL. As one might notice, building blocks of OWL are very similar to DL: concepts map to classes and roles to properties. Unsurprisingly, the OWL Description logic derivate is formally known as OWL DL [29].

2.1.10 Fuzzy Logic

Fuzzy or *multi-valued logic* is determined as a set of mathematical principles for knowledge representation based on degrees of membership rather than on crisp

memberships of classical binary logic [6]. The fundamental concept fuzzy logic builds upon is a *fuzzy set*. The core fuzzy set theory assumption is that an element belongs to a set with a certain degree of membership. A fuzzy set is usually defined by a *membership function* of various shapes which determines the degree of membership of individual elements. Examples of several fuzzy sets can be seen in Figure 2.6.

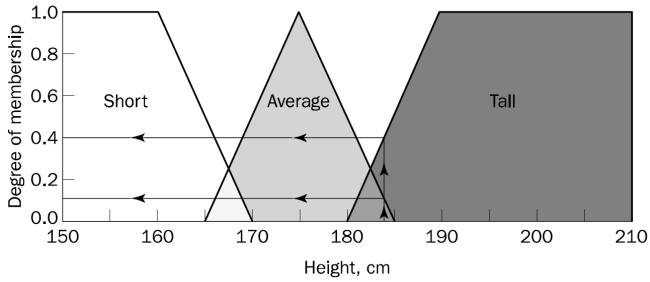


Figure 2.6: Examples of three different fuzzy sets defined on *person height* variable [6].

One of the basic concepts in fuzzy logic are linguistic variables also known as *fuzzy variables*. For example the proposition *"wind is strong"* assigns the linguistic value *strong* to a variable named *wind*. Moreover, fuzzy variables carry with them the concept of *fuzzy set qualifiers*, called *hedges*. Hedges are terms that modify the shape (i.e. boundaries) of fuzzy sets [6]. A few examples of hedges could be: *very, somewhat, less, quite* etc. Several examples of hedges applied to fuzzy sets are shown in Figure 2.7.

As fuzzy sets inherit their base from set theory, standard set operation namely: *complement, containment, intersection* and *union* can be applied to them as well. This fact lies the foundation for *fuzzy rules*. Fuzzy rules form conditional statements in the form: *IF ... THEN ...* and are usually used for capturing human knowledge in fuzzy logic based systems. An example of a fuzzy rule could be *"IF wind is strong AND water is warm THEN sailing is good"*.

In order to reason about fuzzy sets and elicit knowledge from a fuzzy expert system, *fuzzy inference* is used. Fuzzy inference can be defined as a process of mapping from a given input to an output, using the theory of fuzzy sets [6]. There are two commonly used types of fuzzy inference: *Mamdani-style* and *Sugeno-style*.

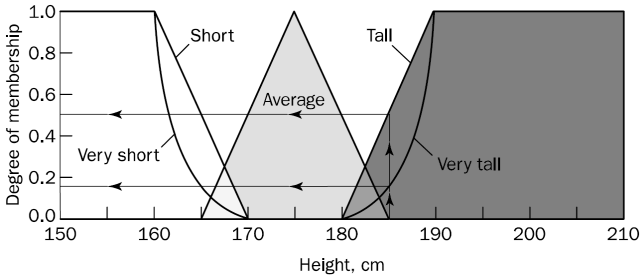


Figure 2.7: Examples of hedges demonstrated on the fuzzy sets from Figure 2.6 [6].

However, both of them are quite similar and usually involve the following four steps: *Fuzzification, Rule evaluation, Aggregation of rule outputs* and *Defuzzification.*

2.1.11 Unity Real-time Development Platform

Unity is a development platform primarily focused on game development. The software consists of a game engine and visual scene editor. For game logic programming, C# scripting language is used. The platform has frequently been used for projects beyond game development ranging from interactive simulations environments or console games to immersive AR/VR experiences. The engine also supports compilation to wide variety of target platforms [31].

2.1.12 Lock-free and Wait-free Concurrency

An algorithm which utilizes concurrency is considered *wait-free* if every access by a non-faulty process is guaranteed a response regardless of whether the other processes are slow, fast or have crashed [32] i.e. it ensures that the response arrives in a finite number of steps [33]. *Lock-free* algorithms guarantee that at least one of the responses arrives in a finite number of steps. Lock-free algorithms are also dead-lock free but can suffer from starvation. Moreover, wait-free algorithms do not suffer from starvation [34].

2.1.13 Rust Programming Language

The Rust language is a multi-paradigm strongly-typed system programming language developed by Mozilla Research. It is characterized by several unique design

principles which provide the language with strong guarantees about isolation, concurrency and memory safety. The result is that the language is essentially free of memory errors as well as of data races [35]. Additionally, the language is very performant and suitable for low-level and efficient software development [36].

The first principle is called *ownership*. In Rust the memory is managed by ownership system which allows compile time checks for possible illegal memory uses. As a result, there is no need for explicit memory allocations or garbage collector. Memory gets allocated when variables are first declared (the memory is then also owned by the variable) and deallocated when the variables happen to go out of their scope.

The second principle introduces *references and borrowing* which are both tightly coupled with the ownership system. Borrowing refers to the use of references i.e. passing by reference. There are two types of references: *mutable* and *immutable* ones. The limitation is that there can be at most one mutable reference to a given piece of memory at a given time and also no immutable references. This fact in connection with the ownership system provides the compiler with enough information for preventing race conditions and dangling references. Rust also has a concept of *lifetimes* which describe for how long a particular reference lives and is safe to use. Lifetimes are also checked during compile time and potential violations result in failed compilation [37].

Moreover, there are more features which are worth mentioning e.g. an enforcement of composition-over-inheritance principle by explicitly disallowing the use of inheritance in the language or convenient zero-cost abstractions.

2.1.14 Text-To-Speech

Dutoit [38] defines *Text-To-Speech* system as a computer-based synthesizer system that should be able to read any text aloud. A very general Text-To-Speech synthesizer typicaly consists of two modules: *Natural Language Processing* and *Digital Signal Processing*. Natural Language Processing module deals with phonetic transcription of the read text while preserving desired intonation and rhythm. Digital Signal Processing module then transforms the symbolic representation it receives into speech (i.e. corresponding spoken waveform).

According to Mache et al. [39], both modules can be further decomposed into several stages. The first stage of Natural Language Processing module is called *Document structure detection* and focuses on interpreting punctuation marks and paragraph formatting. The goal of the next *Text normalization* stage is to handle

abbreviations and acronyms. The last stage performs *Linguistic analysis* which includes morphological analysis for proper word pronunciation and also syntactic analysis to achieve good accenting and phrasing. The Digital Signal Processing module composes of the two following stages. The first stage deals with *Phonetic analysis* and the second one with *Prosodic analysis*.

When it comes to speech synthesis techniques, there are three main categories of them [39]: *Articulator synthesis* which aims to computationally simulate the neurophysiology and biometrics of speech production apparatus, *Formant synthesis* in which individual speech segments are stored on a parametric basis and *Concatenative synthesis* that builds on synthesizing sound by concatenating samples of prerecorded sounds called units. Recently, Text-To-Speech systems based on neural networks have achieved great improvements in the produced speech quality which extends their span to various new domains [40].

2.2 Structured Literature Review Protocol

In the following section, literature and research relevant to the aim of this work is identified through *Structured Literature Review* (SLR). The chapter starts with the identification of relevant sources, proceeds to search strategy description and finally assesses the quality and usability of the found work. Based on the results, the well-suited techniques and knowledge is extracted and applied to the assessment system design. The review was carried out during *Suggestion* phase of *Design Science Research* methodology (see section 1.4).

2.2.1 Identification of Research

This subsection describes how we discover the correct areas of research and obtain suitable keywords for the later *in-depth search*. At first there was an *exploratory literature search* whose purpose was to clarify commonly used terms which appear in connection with the research of our interest. The result of this activity is captured in Table 2.1 with six groups of keywords relevant to our problem domain. Each group represents one area of research we are interested in. The justification and description of each individual group of search terms can be found below the already mentioned Table 2.1.

Group 1	Group 2	Group 3
Data oriented	Multi agent	Real time
Component based	Agent based	Interactive
Entity	Agent	RIS
Component		Simulation
System		
ECS		

Group 4	Group 5	Group 6
Ontology	Rule based	Driver
Semantic graph	Reasoning	Traffic
Knowledge graph	Logic	Road
	Stream	Instructor
		Tutor

Table 2.1: SLR – Search terms and their corresponding groups used during the in-depth literature search

Group 1: The first group steers the focus towards the main concern of this work which is Data-oriented design. Additionally, it also includes terms related to a popular Data-oriented solution which is Entity Component System architectural pattern.

Group 2: The second group further narrows down the search space by including multi-agent systems related literature as the core of the designed system should be agent-based.

Group 3: The third group brings the constraints of real-time execution. The system is required to work in real-time and cooperate with an interactive simulated environment which provides it with input data.

Group 4: This group concerns ontologies and knowledge graphs. There is a need to create an accurate representation of the simulated world which puts a lot of crucial requirements on the candidate data structures. Ontology and knowledge graphs seem to be a promising option for the world abstraction model.

Group 5: The data produced by the assessment system needs to be suitable for logic reasoning on several levels of complexity, therefore, the search should provide supplementary research about logic and reasoning.

Group 6: Finally, the whole work is tightly tied to traffic and driving domain and also concerns driving instructors and virtual tutors. For that reason, the last group includes keywords relevant to this area of research.

The online digital sources listed below were used to search for the relevant literature and basically to conduct the whole structured literature review in general:

- *ACM Digital Library*
- *Engineering Village*
- *Google Scholar*
- *IEEE Xplore Digital Library*
- *Science Direct*
- *Semantic Scholar*
- *Wiley Online Library*

The in-depth search was conducted in two phases called *primary* and *secondary* search as the primary phase revealed only very little relevant research on this topic.

During the primary phase, the search query was constructed from all available search terms of all groups. Basically, all the terms of each individual group were connected by OR operator and groups were connected together by AND operator. The actual query looks as follows:

(Data oriented OR Component based OR Entity OR Component OR System OR ECS) AND (Multi agent OR Agent based OR Agent) AND (Real time OR Interactive OR RIS OR Simulation) AND (Ontology OR Semantic graph OR Knowledge graph) AND (Rule based OR Reasoning OR Logic OR Stream) AND (Driver OR Traffic OR Road OR Instructor OR Tutor).

However, it turned out that the search query is too specific and does not provide enough of the desired results. Hence, it was necessary to conduct several less restricted searches which utilize only parts of the previously shown query. Using just a few of the provided groups of search terms yielded the best results. The list of used combinations is provided below:

- *(Group 1) AND (Group 2) AND (Group 3)*
- *(Group 1) AND (Group 3) AND (Group 4)*
- *(Group 1) AND (Group 2) AND (Group 6)*

- *(Group 2) AND (Group 4) AND (Group 5)*
- *(Group 4) AND (Group 5) AND (Group 6).*

2.2.2 Selection of Primary Studies

The primary studies signify a subset of all the found literature which satisfies certain criteria of relevance. Even though the search query filters out most of the non-relevant literature, it is still not enough to guarantee the required level of relevance.

The results of all searches were, therefore, subjected to inclusion criteria evaluation. Each found source must fulfill at least one of the provided inclusion criteria otherwise it is discarded. Fulfilling more than one criteria is preferred but not explicitly required. This process greatly reduced the amount of found studies to a manageable subset. The Table 2.2 lists all of the used criteria.

ID	Inclusion criteria
IC 1	The study's main concern is Data-oriented design.
IC 2	The study's main concern are multi-agent or agent based systems operating in real-time.
IC 3	The study focuses on real-time simulated environments.
IC 4	The study concerns logic or reasoning.
IC 5	The study concerns ontologies, semantic graphs or knowledge graphs.
IC 6	The study concerns the domain of traffic or car driving.

Table 2.2: SLR – Inclusion criteria table

2.2.3 Quality Assessment

Finally, the quality of the remaining studies was assessed in order to determine preferred and dependable research. It was done so by raking each of the study based on if and to which extent it fulfills each individual quality criteria. The scoring is either 0 (not at all), 0.5 (to some degree) or 1 (clear fulfilment) points. The used criteria are defined and described in Table 2.3. They are mostly of general nature assessing the overall quality of each study from the academic point of view.

ID	Quality criteria
QC 1	The study has a clear statement of the aim of the research.
QC 2	The study is put into context with other studies and research.
QC 3	The study conducts a set of documented experiments and presents their findings.
QC 4	The study contains a discussion of the results.

Table 2.3: SLR – Quality criteria table

The result of quality assessment of all the studies is shown in Table 2.4. The table scores each criteria individually and also shows the total score of each study. The score in general serves more as an indicator of how well were the studies conducted and is not directly scoring their results or findings.

Study	QC 1	QC 2	QC 3	QC 4	Score
Doniec2008	1	0.5	1	0.5	3
Bosse2008	1	0	0.5	1	2.5
Garcia2014	1	0	0	1	2
Gutierrez2014	1	1	0	1	3
Danielsson2015	1	0	0.5	1	2.5
Nguyen1997	1	1	0	0.5	2.5
Morignot2012	1	0.5	0.5	1	3
Toulni2015	1	1	0	0	2
Zhao2015	1	1	1	1	4
Hodson2018	1	0.5	0	0.5	2
Sharp1980	0.5	0.5	0	0.5	1.5
Weiss1998	1	0.5	0.5	0.5	2.5
Hall2014	1	0	0.5	0.5	2
Su2014	1	1	0.5	1	3.5
Lange2016	1	1	1	1	4
Fontana2017	1	1	1	1	4
Fuchs2008	1	1	0.5	0.5	3
Oulhaci2013	1	1	0	0.5	2.5
Sukthankar2002	1	0.5	1	1	3.5
Armand2014	1	1	1	1	4
ZhaoIchise2015	1	1	1	1	4
Mohammad2015	1	1	1	0.5	3.5
Fang2019	1	1	1	1	4
Bermejo2012	1	1	1	0.5	3.5
Buechel2017	1	1	0.5	1	3.5
Krol2013	1	0.5	1	1	3.5
Wang2016	0.5	0.5	0	0.5	1.5
Shoham1987	1	1	0	0	2
Demiryurek2009	1	1	1	0.5	3.5
Hulsen2011	1	1	0.5	0.5	3
Schmalstieg2019	0.5	0	0	0.5	1
Regele2008	1	0.5	0.5	1	3
Kallimanis2016	1	1	0	1	3
LangeWeller2016	1	1	1	1	4
Krotzsch2012	1	0.5	0	0.5	2

Table 2.4: SLR - Results of quality criteria evaluation

2.2.4 Data Extraction

The final subset of studies was subjected to data extraction process. The extracted knowledge for each study is presented in a form of a short summary. The summaries are ordered in the same way as the studies are in Table 2.4.

Doniec2008 – A Behavioral Multi-Agent Model for Road Traffic Simulation

The paper from Doniec et al. [41] describes a multi-agent approach to road traffic simulation with a particular focus on intersections. The important detail is that the agents are based on driver's behavioral model i.e. reflect real-world behaviour of drivers such as overestimation or impatience. The first part of the paper reviews existing approaches to traffic simulation problem. The review includes multi-agent system approach which is also the main focus of this paper. The concern is the coordination of agents at intersections i.e. preventing them from colliding with each other. The proposed solution consists of priority-based rules which in fact mirror yielding rules at intersections. The rules were implemented in ArchiSim simulation tool and validated via several experiments on real intersections using real traffic data.

Bose2008 – A Component-based Ambient Agent Model for Assessment of Driving Behaviour

Bosse et al. [42] proposes agent-based ambient model that addresses the assessment of driving behaviour in order to enhance driver's safety. The system constantly checks the driver via sensors and if unusual behavior is detected it pulls over and stops the car. The system agents are represented as components which enhances modularity. The agents utilize predicates called state ontologies (as a part of their internal processes) and First order logic reasoning to derive their conclusions. The system contains several components/agents types with different responsibilities such as inter-agent interaction or world information management. The whole system was tested in LEADSTO simulation environment.

Garcia2014 – A Data-Driven Entity-Component Approach to Develop Universally Accessible Games

The paper from Garcia and de Almeida Neris [43] presents an Entity Component System (ECS) architecture applied to the domain of Universally-Accessible games. In this paper, the main exploited advantage of ECS is its flexibility and ease of change in order to support players with as many disabilities as possible. Substantial part of the paper compares Object-oriented programming to ECS

architectural pattern and highlights advantages of the latter. Furthermore, the paper describes a data-driven addition to ECS i.e. a possibility to specify component composition via XML files.

Gutierrez2014 – Agent-Based Framework for Advanced Driver Assistance Systems in Urban Environments

In Gutierrez et al. [44] the authors propose a novel safety-focused agent-based high level reasoning system for Advanced Driver Assistance Systems. The system utilizes an ontology as a shared data communication platform which holds the observed state of the world. In addition, manually created rule repository contains the logic for executing actions. The individual agents are divided into listeners (which build ontology) and reasoners (which execute rules from rule repository). The system also takes into account image snapshots of the environment and driver's gaze. The result is demonstrated on a set of several common traffic scenarios such as parking or pedestrian avoidance.

Danielsson2015 – A High Performance Data-Driven, Entity-Component Framework For Game Engines With Focus on Data-Oriented Design

Danielsson and Bohlin [45] contributed with a paper about Entity Component System pattern and Data-oriented design which also partially describes implementation. The main goal is to exploit best of both worlds i.e. modularity and modifiability of ECS and efficiency and speed of Data-oriented design. A simple test case demonstrating cache efficiency is presented an evaluated.

Nguyen1997 – A Multi-Agent Architecture for Situation Awareness

The work of Nguyen [46] describes an architecture of a system which enhances situation awareness of an aircraft crew in real-time. The architecture uses multi-agent paradigm as a framework for implementation of individual independent but cooperating agents. Each of them is an expert system on its own and has a concern that it reasons about. At first, the paper outlines general characteristics and advantages of intelligent agents. Secondly, specific data sources and actual system agents are described. The output of the system is a complete surveillance picture of the aircraft environment.

Morignot2012 – An Ontology-based Approach to Relax Traffic Regulation for Autonomous Vehicle Assistance

Morignot and Nashashibi [47] propose an ontology-based solution for relaxation

of traffic rules in extreme traffic situations such as an overtake of a broken car. The presented comprehensive ontology of traffic domain is implemented as OWL ontology in PROTÉGÉ editor. The editor is also loaded with custom symbolic inference rules for reasoning about the relaxations. The reasoning procedure is based on Description logic. The problem of ontology changes over time is also addressed. The limitations of the proposed approach are discussed too e.g. the lack of uncertainty representation.

Toulni2015 – An ontology based approach to traffic management in urban areas

The work of Toulni et al. [48] presents a solution for dynamic traffic management. The system utilizes an ontology as the main fusion platform for data extracted from VANET (Vehicular Ad-hoc NETwork) messages. The ontology is implemented in OWL and covers vast number of common traffic domain concepts and their properties e.g. a vehicle and its speed, position, type etc. All the concepts are described in detail and discussed thoroughly.

Zhao2015 – An Ontology-Based Intelligent Speed Adaptation System for Autonomous Cars

The Intelligent Speed Adaptation System developed by Zhao et al. [49] describes another real-time ontology-based solution for the traffic domain. The system at hand can be considered an implementation of Advanced Driver Assistance System or a system for autonomous vehicles. Real-time data streams are provided by car sensors in RDF format and saved into several OWL ontologies. Then, SPARQL queries are used to reason about ontology states. The ontology is of three types: map, control and car ontology and all of them are contained within a knowledge base. The inference rules for reasoning about ontology are specified in Semantic Web Rule Language (SWRL). The system is verified by experiments in a simulated world and also on real world data.

Hodson2018 – Application of ECS Game Patterns in Military Simulators

Hodson and Millar [50] present an application of Entity Component System (ECS) architectural pattern to interactive military simulators. The first part of the paper describes the actual comparison of games and interactive simulators in terms of their main similarities and differences. The second part thoroughly explains the main idea behind ECS and finally puts it in context of military simulators.

Sharp1980 – Data-oriented Program Design

The paper from Sharp [51] brings an overview of Data-oriented design principles. It also presents data flow model of parallel computation based on these principles. Firstly, the paper covers traditional program specification and development methods which are mildly criticized. Secondly, an alternative method based on data flow (instead of more traditional control flow) is introduced. Since then the program is considered as a transformation which takes some given input and produces the desired output. For a precise data flow specification, a custom domain specific language describing data flow and dependencies is designed and presented. A few examples of the language usage are presented e.g. parallel merge sort description.

Weiss1998 – Design and implementation of a Real-time Multi-agent system

The work of Weiss and Steger [52] shows an experimental distributed multi-agent system (VEX) for real-time fault diagnosis. The proposed system combines methods from AI and real-time systems fields. Each agent is a separate computer system consisting of two components: reflexive component (handles the real-time aspect, responds to stimuli etc.) and cognitive component (which is the actual problem solving part). The system also utilizes a simulator in order to gather expected states and values for comparison with the real world data. There is also a knowledge model involved which encodes information about possible fault causes. In case of discrepancies between simulator and real system output, the knowledge model is consulted about possible fault causes. As the system is distributed, its implementation is based on message passing.

Barber2005 – Design, Runtime, and Analysis of Multi-Agent Systems

Barber et al. [53] offer a study about available analysis and design tools suitable for multi-agent systems development. Several tools are presented and discussed such as TPM which allows designers to compare available technologies based on required agent competencies or Tracer tool whose purpose is to analyze run-time agent data (i.e. agent comprehension or debugging). The use of tools is demonstrated in Unmanned Aerial Vehicle (UAV) target tracking simulation.

Hall2014 – ECS Game Engine Design

In [54] the authors explore common game engine design approaches and also

develop a custom modular solution based on existing Entity Component System (ECS) frameworks. At first, the paper compares Object oriented Programming (OoP) with ECS pattern. The weaknesses of OoP, especially the difficulty of hierarchy changes and the problem of expandability, are discussed. In contrast, composition over inheritance philosophy of ECS addressing these issues is presented. Additionally, disadvantages of ECS e.g. inter-system communication are also outlined. The two covered ECS frameworks are Cupcake ECS and Artemis ECS. Both of them are thoroughly analyzed and their main shortcomings are pointed out. The best of both frameworks was taken and forged into a custom developed ECS solution.

Su2014 – From a Link-Node-Based Network Representation Model to a Lane-Based Network Representation Model: Two-Dimensional Arrangements Approach

The paper from Su et al. [55] describes a road network representation based on lane data and also proposes an approach capable of generating this kind of representation from publicly available datasets. The lane based representation allows more granular reasoning about a road network and is, therefore, very well suited for simulated environments based on real locations. At first, the paper outlines advantages and the needs for lane based representation. Then, it compares lane based representation to more common link-node representation. Several proposed ways of converting the latter into the former and their issues are discussed. Nextly, a custom conversion solution based on 2D arrangements is introduced. The solution does not work with generic 2D arrangements thus three specific customizations are introduced. Then, each step of the conversion process is explained in detail. Finally, the algorithm and its implementation is verified and evaluated on two sample road networks.

Lange2016 – GraphPool: A High Performance Data Management for 3D Simulations

The work of Lange et al. [56] presents a novel approach for simulation state management. The solution is called GraphPool and combines wait-free hash maps with graph structures. Furthermore, it supports data queries similar to those found in relational databases. Firstly, the solution is compared to standard relational databases and the most limiting factors of using the database approach in simulations are outlined. The proposed solution works in-memory and is schema-less which further helps it to prevail over relational databases. Another advantage lies in the system's ability to handle concurrent data access in a wait-free manner. The system consists of three main components: Con-

currency control management, Object-oriented data store and Relational query engine. Moreover, each system component possesses its own local state that is on demand synchronized with the global world state. The system data is stored in hash maps as data packets (with individual keys for the whole packet and its members) which resembles Data-oriented design principles. This whole structure is aliased as GraphNode. GraphNodes are stored in the central GraphPool object in a graph-like structure. The GraphNodes are timestamped and all their past versions are stored for potential future state replay. The GraphPool also utilizes caching in order to speed up future queries. Finally, the whole solution is implemented in C++ and evaluated on spaceflight simulation case study. The implementation outperformed its competitors by several orders of magnitude.

Fontana2017 – How Game Engines Can Inspire EDA Tools Development: A use case for an open-source physical design library

In [14] Fontana et al. present application of Data-oriented design principles into the field of electronic design tools. These tools, similarly to game engines, need to be able to handle huge amounts of data while still remain fast and interactive. The paper starts with common comparison of Object-oriented design (OoD) and Data-oriented design (DoD). The main advantages of DoD are outlined and also the way how it overcomes main shortcomings of OoD design are discussed. The final solution utilizes Entity Component System architectural pattern which is one of the feasible ways of implementing DoD. In the evaluation part two system prototypes were implemented and tested. The results clearly showed faster execution times and lower latency of DoD in the first test and comparable performance in the second test. In addition, the custom ECS implementation was extracted to an open-source library called Ophidian.

Fuchs2008 – Integration of Ontological Scene Representation and Logic-Based Reasoning for Context-Aware Driver Assistance Systems

Fuchs et al. [57] focus on design and sharing information among cooperative driver assistance systems. The paper first introduces an ontology context model for driving scene description. The ontology was developed in OWL and covers vast number of concepts from driving domain such as traffic signs, lanes or nearby cars which gives it the ability to accurately describe the driving scene at hand. However, there still might be uncertainties in the data inputs. Each concept or relation contains meta information which specify information such as certainty, source reliability, expected time-span etc. Spatial information is represented relatively to the Ego car of interest. Traffic rules are introduced and described in Semantic Web Rule Language (SWRL). Some of the rules were implemented as

constraints, therefore, constraint satisfaction could be used to derive conclusions about the current traffic situation. The workings of the system were successfully demonstrated on an overtake scenario where the system assists and advises a real driver.

Oulhaci2013 – Intelligent Tutoring Systems and Serious Game for Crisis Management: A Multi-Agents Integration Architecture

Oulhaci et al. [58] present work about Serious Games (SG) an their usage together with Intelligent Tutoring Systems (ITS) in crisis management training. At first, the study defines the concept of SG and evaluates existing work about it. This part also introduces SIMFOR, a multi player game which allows various stakeholders to practice a chosen crisis management skill. The game support collective as well as individual decision making and evaluation. The goal of the paper is to integrate several functionalities common in ITS into the SIMFOR game. The focus is on three modules: Learner module (used for understanding the learner and adaptation), Expert module (contains expert knowledge) and Pedagogical module (selecting the most appropriate knowledge and schedule exercises). The proposed system implementation utilizes cooperative multi-agent architecture to at first simulate human players (Belief Desire Intention agents) and implement ITS functionalities and metrics (evaluation agents). The domain knowledge of the system is stored in an ontology. Lastly, the system is demonstrated on an emergency scenario defined by domain expert.

Sukthankar2002 – Multiple Adaptive Agents for Tactical Driving

Sukthankar et al. [59] explore the possibilities of using multi-agent systems for reasoning about tactical driving (i.e. determining maneuvers and other short term goals). The developed multi-agent system is called SAPIENT. Each reasoning agent in it is modelled as an independent entity which reasons about a subset of sensor data coming from perception modules. Each agent votes for a possible action to take, the votes are accumulated and the most voted action is executed. The interactions between agents, their influence of final decision and their other internal states parameters are automatically tuned by evolutionary optimization strategy called Population-Based Incremental Learning (PBIL) which makes the agents able to learn automatically. The system was compared with rule-based alternative and further evaluated in automated highway driving test. Two scenarios were tested, namely overtake and safely taking a highway exit.

Armand2014 – Ontology-Based Context Awareness for Driving Assistance Systems

The work of Armand et al. [60] focuses on enhancing driving space awareness of the subject's vehicle ADAS. The proposed solution uses sensors to track and interpret spatio-temporal relationships between Ego and its environment in order to predict behaviour of other nearby entities. The solution is based on OWL ontology which captures and stores the context information. The ontology concepts are divided into two classes: *Mobile entities* (vehicles, bikes...) and *Static entities* (traffic signs, roads, pedestrian crossings...). The ontology also stores *Context parameters* i.e. relations between Static and Mobile entities. The inference system is rule based and the rules are specified by hand in SWRL language. For the reasoning part Description logic and Pellet reasoner is employed. The correct behaviour of the system was evaluated in real-time on pedestrian detection scenario.

ZhaoIchise2015 – Ontology-based Decision Making on Uncontrolled Intersections and Narrow Roads

In [61] Zhao et al. describe application of an ontology-based decision system to the specific areas challenging for both autonomous vehicles and ADAS. As the name suggests, the paper mainly focuses on two of them: narrow roads and uncontrolled intersections. The system gathers data from sensors which are saved as RDF data streams and potentially accessed by SPARQL query engine. The employed OWL ontology is of three types: *Map ontology* (describes road network), *Control ontology* (captures main vehicle's current state and path) and *Car ontology* (defines all other cars involved in the situation at hand). The if-then reasoning rules are defined in SWRL and on-demand processed by a reasoner. The rules are almost exclusively specifying right of way or collision avoidance. Two experiments were conducted: one in simulated environment and one on real car in real traffic. The system has proven the ability to effectively command vehicle's path planning system in order to avoid collisions and successfully handle the challenging traffic situations.

Mohammad2015 – Ontology-based framework for risk assessment in road scenes using videos

The paper by Mohammad et al. [62] explores feasibility of real-time ontology construction from a live video stream. Based on the inferred facts, the presented system calculates potential risks in the current driving situation. As sole detection of entities in a video stream is not sufficient for the complete situation understanding, an ontology capturing the scene concepts and relations is employed. It consists of three main classes which all represents factors contributing to the final risk. The classes are: *Risk emerging* from object/entity, *Environ-*

mental risk and *Road environmental* risk. The final level of risk is inferred by manually specified rules. SPARQL language was used for querying the ontology. The evaluation of the system focused on risky situations involving pedestrians. YouTube videos with pedestrians were used as a test data source. The evaluation proved correct and accurate risk assessment.

Fang2019 – Ontology-based Reasoning Approach for Long-term Behavior Prediction of Road Users

The work of Fang et al. [63] focuses on long-term drivers behaviour prediction. It employs an ontology which supplies conceptual situation description and Markov Logic Network (MLN) for inferring likely behaviour of other road users, therefore, addressing uncertainty. The proposed framework is made up of three modules: *Situation generator*, *Situation representation* and *Scenario generator*. Situation generator detects and generates relationships between Ego vehicle, other surrounding objects and map objects. Situation representation infers the probability of candidate intentions of each situation object. Ontology is used as a knowledge representation method. Scenario generator is split into two modules: *Behaviour reasoner* and *Timing generator*. Behaviour reasoner is used to infer occurrence probability of candidate intentions in the next time step. MLN specifies the rules which guide the inference process. Timing generator helps to assign probabilities to other possible future states ahead. The framework was tested in real-time under ideal conditions while driving through intersections.

Bermejo2012 – Ontology Based Road Traffic Management

Bermejo et al. [64] propose an integration of ontology into vehicle systems in order to provide them with reasoning capabilities. The paper is focused on allowing emergency vehicles to pass through traffic safely. The proposed ontology structure is derived from A3ME ontology. It basically extends the base A3ME ontology in terms of additional concepts as well as additional rules. During driving, the ontology is populated with sensor data in real-time. If a potential risk event is detected, alarm is broadcasted to all other emergency vehicles on the road. Drivers can then take advantage of suggestions provided on a screen inside of a car. The evaluation was performed in a simulation and focused on safe overtakes performed by emergency vehicles.

Buechel2017 – Ontology-Based Traffic Scene Modeling, Traffic Regulations Dependent Situational Awareness and Decision-Making for Automated Vehicles

The paper from Buechel et al. [65] presents a framework for traffic regulation based decision-making. The focus of the framework is on autonomous vehicles. For the description of traffic concepts and relations between them OWL2 ontology is used. The ontology contains comprehensive description of the scene especially in terms of the road network. The reasoning part uses Description logic encoded traffic rules specified in SWRL and utilizes Pellet reasoner. The decisions are derived in real-time directly from the stored traffic rules and current ontology state. The framework is built from individual modules which allow easy potential adaptations to different sets of traffic regulations of different countries. The solution is evaluated and validated on several scenarios: 4-way controlled intersection, 4-way uncontrolled intersection, 4-way uncontrolled intersection with tram tracks and police officer controlled intersection.

Krol2013 – Practical Performance Aspects of Using Real-Time Multi-Agent Platform in Complex Systems

Krol and Nowakowski [66] investigate performance aspects of real-time multi-agent systems. The paper focuses on multi-agent system implementation in Java (JADE) which deals with vehicle control and complies to real-time specifications. The system is called CARS. Furthermore, the implementations addresses concepts of multi-threading and distributed execution which implied custom implementation of thread management and scheduling. The goal of the evaluation part was to prove that the employed performance enhancement techniques speed up the system. The implementation was tested in two synthetic benchmarks. The main measured performance properties were thread and agent delays. However, the experiments showed weaknesses of the current implementation, mainly due to JADE framework limitations.

Wang2016 – Study of Semantic Reasoning based on Ontology Description Logic

Jinhuan Wang and Baomin Li [67] conducted a study about feasibility of Description logic (DL) reasoning about ontologies. Ontology and its construction for an example fruit domain (Apple-Onto) serves as a basis for semantic reasoning discussions. At first, the paper describes OWL language and its reasoning capabilities. Then, it defines several ontology construction principles which improve the overall quality of developed ontologies. The notions of DL TBox and ABox are explained and so is reasoning based on both concepts. The reasoning performed on Apple-Onto confirmed the possibility of reasoning about several facts e.g. classification of entities or ontology data consistency checking.

Shoham1987 – Temporal Logics in AI: Semantical and Ontological Considerations

Shoham [68] explores representation of temporality aspects within logic languages and temporal reasoning in general. The work primarily focuses on two most influential formalisms proposed by Allen and McDermott called Interval calculus and Temporal logic respectively. In both of them, several inconsistencies were found. Both formalisms are, therefore, compared, their similarites are identified and utilized as a basis for a new improved formalism. The new formalism is termed Interval logic and proposes solutions to the identified issues.

Demiryurek2009 – Towards Modeling the Traffic Data on Road Networks

The work of Demiryurek et al. [69] concerns spatio-temporal networks and their accurate modeling. The paper proposes a framework for modelling spatio-temporal traffic networks which dynamically capture the times needed to travel through individual segments based on traffic flow. The framework is based on real world historical traffic data. The modelling approach consists of three steps: computing time-dependent travel times from historical data, labelling individual regions based on their spatial characteristics and grouping similar traffic flows into respective spatial characteristics. The purpose is to find the most representative traffic flows within network regions. The framework was successfully validated by conducting experiments with several different road networks. Furthermore, a spatio-temporal road network model of Los Angeles county was generated and open-sourced.

Hulsen2011 – Traffic Intersection Situation Description Ontology for Advanced Driver Assistance

In [70] Hulsen et al. present an approach to generate complete traffic situation description for Advanced Driver Assistance Systems (ADAS). The basis for situation description is an OWL ontology which is validated and further enhanced by reasoning about traffic rules. The first part of the paper introduces theory and related research necessary for a good understanding of the proposed approach. As the paper focuses on intersections, the ontology covers only this part of the traffic domain. The ontology, however, supports loading of additional parts as modules, therefore, it is capable of dynamic expansion. The used rule set encodes traffic regulations hence allows more precise Description logic based reasoning. The ontology is queried by intelligent agents who extract desired information for ADAS. The real-time aspect of the proposed system is beyond its current capa-

bilities. A single reasoning query can take up to several seconds which does not meet real-time criteria. The work is demonstrated on several examples which focus on reasoning about intersections. Both controlled and uncontrolled intersections are considered.

Schmalstieg2019 – Unified Patterns for Realtime Interactive Simulation in Games and Digital Storytelling

The paper from Schmalstieg [71] discusses the use of several architectural and software design patterns in interactive simulations and game engines. The purpose of the study is to show that these design patterns share a common ground and can be freely combined without introducing any overhead. The paper introduces several concepts that gave rise to variety of common design patterns, namely notifications, dataflow, publish/subscribe, Model View Controller and Entity Component System (ECS). Each of them is discussed separately. In the end, the paper proposes a combined pattern called Brokered ECS which is based on ECS, publish/subscribe and dataflow. The Brokered ECS is suggested as an ideal choice for simulations and games.

Regele2008 – Using Ontology-based Traffic Models for more efficient Decision Making of Autonomous Vehicles

Regele [72] describes application of high-level world model to the domain of autonomous driving systems. The ontology-based model consists of low-level part dedicated to trajectory planning and high-level part which reasons about traffic coordination. Both of the parts are handled by separate systems. The work primarily focuses on reasoning about situations at intersections. The road network is modelled as a graph-like structure of connected lanes. Nearby vehicles and other objects related to lanes have their positions specified within them. Relations between individual lanes make up the whole road network. The decision-making process is rule based and the rules are specified in advance. The proposed system was used within CyberCars2 project and was proven to be helpful for high-level reasoning of autonomous cars.

Kallimanis2016 – Wait-Free Concurrent Graph Objects with Dynamic Traversals

The paper from Kallimanis and Kanellou [73] proposes a wait-free concurrent graph data structure model capable of atomic snapshots. The model is later implemented as an adjacency matrix based graph structure named Dense. The implemented structure supports complete and partial dynamic traversals and ad-

dition or removal of edges or vertices. However, the number of vertices needs to be known in advance (or at least their maximal count). The results is validated by theoretical proof of correctness without supplying any empirical evidence.

LangeWeller2016 – Wait-Free Hash Maps in the Entity-Component-System Pattern for Realtime Interactive Systems

In [74] Lange et al. introduce high-performance lock-free and wait-free hash map data structure for real-time interactive systems (simulations, computer games etc.). The intended use case for the hash map is storing components in Entity Component System architectural pattern. The basic idea behind the wait-free nature of the data structure is double buffering of the data it contains. Every component stores two versions of itself: *producer* and *consumer* version. When a System tries to write to a Component it receives a clone of the current producer version. After applying modifications to it, the newly written producer version is returned to the component and marked as a new producer and consumer version as well. The old consumer versions are kept in memory until all reads from them are finished, then they are discarded. The paper proposes different schemes for storing versions of consumer component data based on responsibilities for their deallocation. Two approaches are proposed: centralized (component is responsible) and decentralized (system which reads from it is responsible). The different variants of the memory management are empirically evaluated in spacecraft simulation engine. All implementations, however, outperformed lock-based approaches.

Krotzsch2012 – A Description Logic Primer

The paper from Krötzsch et al. [29] provides a formal introduction to Description logic (DL) and reasoning. At first, the paper explains how DL provides means for modelling the relationships between entities in a domain of interest. Nextly, the basic building blocks of DL i.e. concepts, roles, constructors, *ABox* and *TBox* axioms are introduced. The DL family $\mathcal{SROIQ}$ is further explained in detail. Finally, a relation of DL to modern OWL is discussed.

2.2.5 Data Synthesis

The data extraction phase yielded considerable amount of research for the vast majority of the fields of our interest. The following paragraphs summarize the information found and put them in context with the goals of this work.

Data-oriented design was first introduced and discussed in the paper from Sharp

[51] which is considered the cradle of DoD. The papers from Fontana et al. [14] and Danielsson and Bohlin [45] put DoD into practice and even connect it with ECS architectural pattern and the usage in performance-intensive applications. When it comes to bare ECS, several papers are concerned about this topic (Hodson and Millar [50], Hall [54], Schmalstieg [71], Lange et al. [74]). Most of these, utilize the pattern in simulated environments, computer games or other real-time or performance critical environments.

Multi-agent systems are also frequently used for data extraction and processing in the traffic domain. Majority of the papers utilize collaborative multi-agent systems (Doniec et al. [41], Bosse et al. [42], Gutierrez et al. [44], Nguyen [46], Barber et al. [53], Oulhaci et al. [58], Sukthankar et al. [59]) whose goal is usually to assess and comprehend a traffic situation at hand. Ontologies and knowledge graphs commonly appear together with multi-agent systems and serve as knowledge base and/or communication and data exchange platform (Morignot and Nashashibi [47], Toulni et al. [48], Zhao et al. [49], Fuchs et al. [57], Armand et al. [60], Zhao et al. [61], Mohammad et al. [62], Fang et al. [63], Bermejo et al. [64], Hulsen et al. [70], Regele [72]). Some of the papers use ontologies primarily for road network representation (Su et al. [55], Buechel et al. [65], Demiryurek et al. [69]).

Various details about Description Logic and reasoning are covered by Jinhuan Wang and Baomin Li [67], Shoham [68] and Krötzsch et al. [29]. Finally, several papers also specialize solely on the real-time aspect in context of the previously mentioned terms (Weiss and Steger [52], Lange et al. [56], Krol and Nowakowski [66], Kallimanis and Kanellou [73]).

2.3 Motivation

The Structured Literature Review described in section 2.2 identified new additional motivation for the research in context of this work. Based on the evidence found in literature, there are no existing multi-agent frameworks that utilize Data-oriented design principles or Entity Component System architectural pattern. Moreover, almost none of the papers concerns real-time communication with some existing simulation environment in order to exchange information. These facts introduce a new dimension to our research, strenghten the motivation for this particular kind of research and enhance potential outcoming value of this thesis.

Chapter 3

System Design and Implementation

This chapter presents the design and architecture of the implemented solution. The following text is split into four sections which describe the main building blocks that make up the whole system. The first section talks about simulated world abstraction, the second section about the design and architecture of the assessment system, the third section is concerned about interfacing with the existing simulator and the last section presents the developed intelligent agents for testing and functionality demonstration. The work presented in this chapter maps to *Suggestion* and *Development* phases of *Design Science Research* methodology (see section 1.4).

3.1 Dynamic Model of Simulated Environment

The first important problem to investigate and solve was the representation of the simulated environment in order to obtain *Situation Awareness* (2.1.5). *Situation Awareness* is needed for the assessment system to enable correct reasoning about traffic situations. Based on the comprehensive work done by Armand et al. [60], Zhao et al. [61], Mohammad et al. [62], Fang et al. [63], Buechel et al. [65], Hulsen et al. [70] and Toulni et al. [48], it was decided to employ a dynamic ontology (see 2.1.6) evolving over time which serves as a sound abstraction of the simulated world.

3.1.1 Traffic Situation Ontology

The proposed *Traffic Situation Ontology* combines several design decisions proposed by various researchers. The idea of separating world entities into *Static Objects* (i.e. objects that do not move) and *Dynamic Objects* (i.e. movable objects) was taken from the work of Fang et al. [63]. The employed approach for relating cars to each other based on their relative position was originally proposed by Hulsen et al. [70]. Assignment of dynamic properties with over time changing values to ontology classes was inspired by Matheus et al. [4]. Finally, the road network representation is based on the work of Morignot and Nashashibi [47]. The resulting ontology is shown in Figure 3.1. All of the ontology classes and corresponding properties are listed in Table 3.1 and all of the relations between them are described in Table 3.2. For the creation and iterative development of the ontology, OWL (2.1.6) and Protégé editor v5.5.0 was used [75].

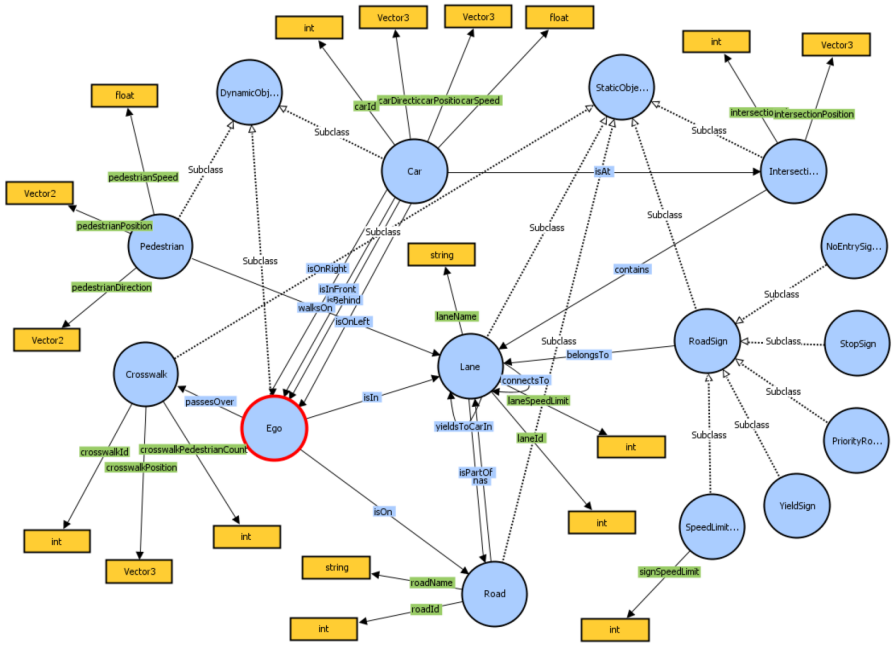


Figure 3.1: Ontology for the description of traffic situations. Dashed arrows define subclassing while filled arrows stand for normal relations. Classes are represented by blue circles and properties by yellow rectangles.

The first ontology class, *DynamicObject*, contains three subclasses: *Ego* which represents the car that is being driven, *Car* which represents all the other cars that are part of the simulated world and *Pedestrian* class that represents all kinds of pedestrians. Regarding *StaticObject* there are four different subclasses with self-explanatory names: *Crosswalk*, *Intersection*, *Lane*, *Road* and *RoadSign*. In addition, *RoadSign* is a superclass for five different types of road signs: *NoEntrySign*, *PriorityRoadSign*, *SpeedLimitSign*, *StopSign* and *YieldSign*.

Class	**Properties**	**Parent Class**
DynamicObject	-	-
Ego	-	*DynamicObject*
Car	carId: int, carSpeed: float, carPosition: Vector3, carDirection: Vector3	*DynamicObject*
Pedestrian	pedestrianSpeed: float, pedestrianPosition: Vector2, pedestrianDirection: Vector2	*DynamicObject*
StaticObject	-	-
Crosswalk	crosswalkId: int, crosswalkPosition: Vector3, crosswalkPedestrianCount: int	*StaticObject*
Intersection	intersectionId: int, intersectionPosition: Vector3	*StaticObject*
Lane	laneId: int, laneName: string, laneSpeedLimit: int	*StaticObject*
Road	roadId: int, roadName: string	*StaticObject*
RoadSign	-	*StaticObject*
NoEntrySign	-	*RoadSign*
PriorityRoadSign	-	*RoadSign*
SpeedLimitSign	signSpeedLimit: int	*RoadSign*
StopSign	-	*RoadSign*
YieldSign	-	*RoadSign*

Table 3.1: The list of classes which form the traffic situation ontology. The ontology is shown in Figure 3.1.

The purpose of different relations is quite transparent, therefore, their detailed description is omitted. However, it is important to note that relations are always directional. In addition, there is a group of relations that might be worth some extra attention. It is the group concerning relations between *Ego* and *Car* (i.e. all the other cars) which consists of four relations: *isInFront*, *isOnRight*, *isBehind*, *isOnLeft*. These relations play an important role in the actual assessment process (as described in later chapters).

Relation	**Properties**	**From Class**	**To Class**
belongsTo	-	*RoadSign*	*Lane*
connectsTo	-	*Lane*	*Lane*
contains	-	*Intersection*	*Lane*
has	-	*Road*	*Lane*
isAt	-	*Car, Ego*	*Intersection*
isBehind	-	*Car*	*Ego*
isIn	-	*Car, Ego*	*Lane*
isInFront	-	*Car*	*Ego*
isOn	-	*Car, Ego*	*Road*
isOnLeft	-	*Car*	*Ego*
isOnRight	-	*Car*	*Ego*
isPartOf	-	*Lane*	*Road*
passesOver	-	*Ego*	*Crosswalk*
walksOn	-	*Pedestrian*	*Lane*
yieldsToCarIn	ifDirection	*Lane*	*Lane*

Table 3.2: The list of relations the traffic situation ontology includes. The ontology is shown in Figure 3.1.

3.1.2 Data-oriented Knowledge Graph

In order to represent the traffic situation ontology in data-oriented manner and allow its dynamic and efficient updates during runtime, a custom data structure called *Knowledge Graph* was designed (see subsection 2.1.6). The data structure is a graph-based structure which was built with four essential data-oriented design practices in mind (for more details see subsection 2.1.2): *DBMS-like memory layout*, *Linear and continuous data structures*, *Data packing and sorting* and *Hot/cold splitting*. Moreover, the data structure is capable of storing all of its previous states which can be recalled on demand at any time. These properties make the structure suitable for back in time reasoning about traffic situations.

The most important part of the data structure is its index substructure which exists in two versions: one for *Knowledge Graph* nodes called *Node Index* and one for *Knowledge Graph* relations correspondingly called *Relation Index*. Both indices are represented as linear arrays which contain metadata about the stored nodes and relations. The indices are meant to separate metadata from the actual data and speedup the lookup of records (*Linear and continuous data structures* and *Data packing and sorting* principles). Moreover, their memory layout is cache friendly as the records are ordered by their timestamps and the most common operation performed on them is sequential lookup. The actual node/relation values are stored in separate heterogeneous linear arrays which are accessed through the mentioned index substructures (*Linear and continuous data structures* and *Hot/cold splitting* principles). The complete memory layout is shown in Figure 3.2. As the Figure illustrates, each *Node Index* record contains:

1. **Timestamp:** Timestamp of the moment when this record and corresponding node value was saved to memory.

2. **Value id**: Index to another linear heterogeneous array which contains the actual values of nodes.

3. **Instance id**: Identifier that allows the graph structure to hold more instances of the same type (e.g. there might several *Car*s surrounding *Ego*)

4. **Outgoing relations index**: An array of indices to *Relation Index* array. The indices point to relations which originate from the node that this *Node Index* represents.

5. **Incoming relations index**: An array of indices to *Relation Index* array. The indices point to relations which end in the node that this *Node Index* represents.

6. **Marker**: A label that defines the type of this *Node Index* record. The record can either signify addition of a new node (*valid* type) or also the deletion/expiration of an existing one (*removed* type). Explanation of this concept with an example is provided later in this subsection, see Figure 3.5.

Similarly, each *Relation Index* record contains:

1. **Timestamp:** Timestamp of the moment when this record and corresponding relation value was saved to memory.

2. **Value id**: Index to another linear heterogeneous array which contains the actual values of relations.

3. **Instance id**: Identifier that allows the graph structure to hold more instances of the same type (e.g. there might several relations to *Car*s surrounding *Ego*)

4. **From node index**: An index to *Node Index* array. The index points to a node from which the relation, that this *Relation Index* represents, originate.

5. **To node index**: An index to *Node Index* array. The index points to a node in which the relation, that this *Relation Index* represents, ends.

6. **Marker**: A label that defines the type of this *Relation Index* record. The record can either signify addition of a new relation (*valid* type) or also the deletion/expiration of an existing one (*removed* type). Explanation of this concept with an example is provided later in this subsection, see Figure 3.5.

Both the *Node Index* and *Relation Index* arrays are enclosed in outer arrays which classify them based on the type of nodes/relations they store (*DBMS-like memory layout* principle). This distinction further speeds up the lookup of nodes/relations because it allows immediate exclusion of the records of different types. The memory layout for *Node Index* is illustrated in Figure 3.3 and for *Relation Index* in Figure 3.4. The capacity of *Node/Relation Index* arrays as well as the enclosing arrays is set beforehand i.e. cannot be increased during runtime. The main reason for compile time fixed capacity are limitations imposed by shared memory regions as further discussed in subsection 3.3.4.

The description of *Node Index* and *Relation Index* substructures has already briefly touched the concept of labelling/marking individual index records. This technique allows the data structure to hold all its previous states and also enables efficient querying of them based on their timestamps. One might have already noticed that the data are actually never deleted from *Knowledge Graph* as they always needs to be available for potential queries. In fact, the amount of data *Knowledge Graph* can store is limited only by its set capacity and available

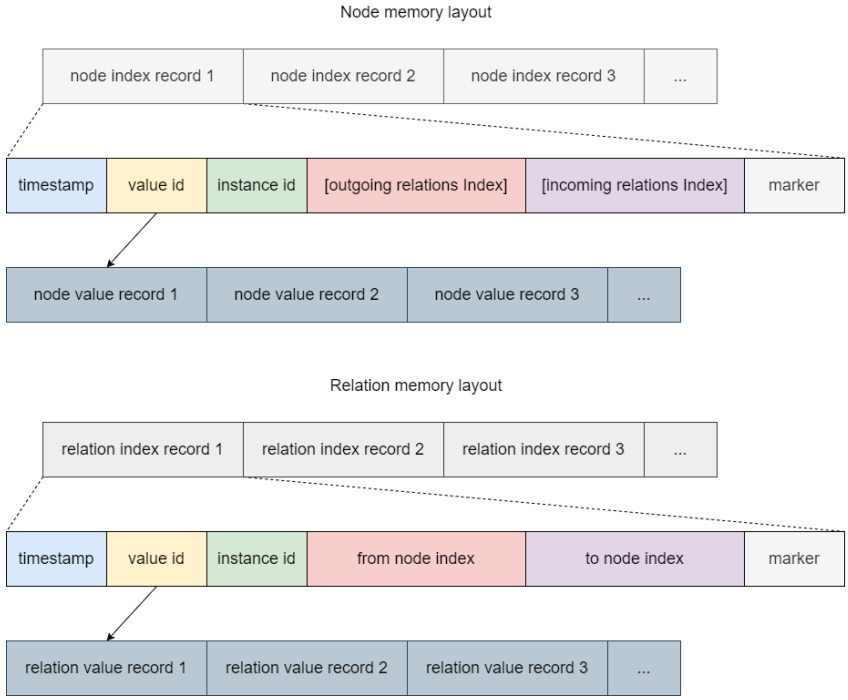


Figure 3.2: The memory layout of *Node Index* and *Relation Index* substructures.

memory. An example of how labelling works is demonstrated in Figure 3.5. An explanation with several *Node Index* records follows below (the explanation applies to *Relation Index* records as well). The first *Node Index* record appears at $t = 2$. Its marker is set to *valid* which means addition of a new node to *Knowledge Graph*. At $t = 4$, there is a new record again which shadows the record from $t = 2$. Therefore, the record from $t = 4$ takes over and is the active one since $t = 4$. However, at $t = 5$ there is a new record again with *removed* marker. This record marks the removal of the node from *Knowledge Graph*, hence the node the record represents ceases to exist. Later at $t = 8$, a new *Node Index* record appears again. The record has *valid* marker, thus signifies an addition of a new node to *Knowledge Graph*.

The next design concept which guarantees the lock-free and wait-free (for more information see subsection 2.1.12) nature of *Knowledge Graph*, is the employed *Graph Indices* substructure. As you can see in Figure 3.6, the *Graph Indices*

Node index storage memory layout

type 1 id	node index record 1	node index record 2	node index record 3	...
type 2 id	node index record 1	node index record 2	node index record 3	...
type 3 id	node index record 1	node index record 2	node index record 3	...
...	...	...	...	...

Figure 3.3: The classification of *Node Index* arrays based on the actual type of node values they represent.

Relation index storage memory layout

type 1 id	relation index record 1	relation index record 2	relation index record 3	...
type 2 id	relation index record 1	relation index record 2	relation index record 3	...
type 3 id	relation index record 1	relation index record 2	relation index record 3	...
...	...	...	...	...

Figure 3.4: The classification of *Relation Index* arrays based on the actual type of relation values they represent.

substructure in fact exist in two copies. The currently active *Graph Indices* substructure is represented by *current indices index* (either 0 or 1) and determines tails for all preallocated arrays within *Knowledge Graph* data structure. When new data for *Knowledge Graph* arrives, they are written beyond the current tails and the new tails are written to currently inactive *Graph Indices*. Finally, the *current indices index* is atomically changed to its new value (either from 0 to 1 or from 1 to 0) which activates the new tail values. This approach makes *Knowledge Graph* safe to modify from one thread and safe to read from multiple threads at the same time. *Graph Indices* also contain a timestamp which marks the moment when they were written.

The basic operations *Knowledge Graph* supports are: *addition of a new node*, *update of an existing node*, *removal of an existing node*, *addition of a new relation*,

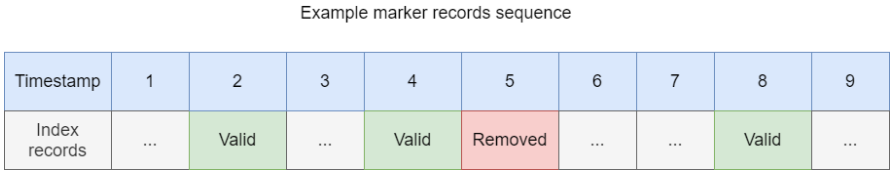


Figure 3.5: An example sequence of several index records with various markers.

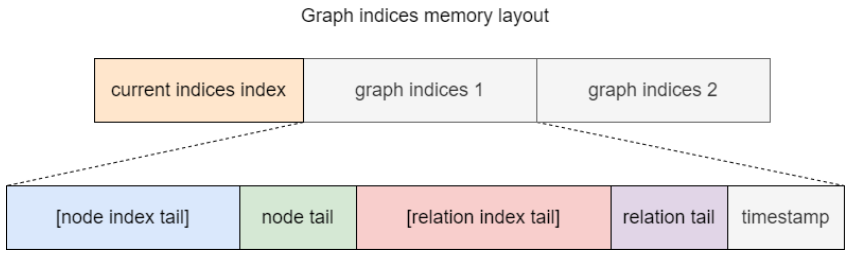


Figure 3.6: The layout of *Knowledge Graph* indices which determine the range of available records/values and also make *Knowledge Graph* data structure lock-free and wait-free.

update of an existing relation, *removal of an existing relation*. The computational complexity of all operations is $O(1)$. Furthermore, there is a possibility to access the raw data directly or through iterators. However, the system uses more sophisticated solution based on *Description logic* (see subsection 2.1.9) described later in subsection 3.1.5.

3.1.3 Transformation and Code Generation

In order to make *Traffic Situation Ontology* compatible with *Knowledge Graph*, code generation practices were employed (see subsection 2.1.7). Code generation separates the ontology design from the actual implementation and therefore permits its easy conversion to various programming languages. This aspect gets particularly important as part of the assessment system resides in its own application implemented in Rust (2.1.13) while another part lives inside of Unity development platform (2.1.11). The custom code generation pipeline runs at compile-time and is illustrated in Figure 3.7.

The first generated product are types representing *Traffic Situation Ontology*

classes and relations which are also compatible with *Knowledge Graph*. The type representations are generated for Rust as well as C# language. Apart from the type conversion, the pipeline also generates API for accessing *Knowledge Graph* from Rust and C#.

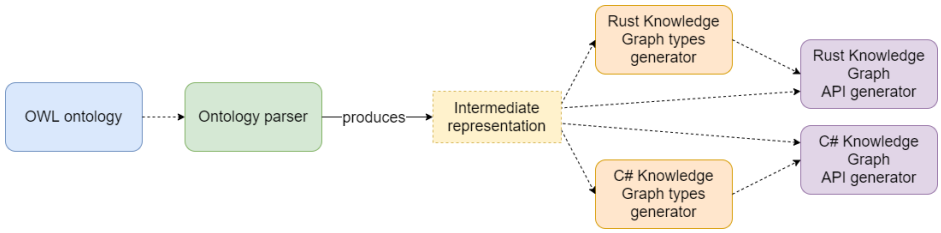


Figure 3.7: The code generation pipeline which converts *Traffic Situation Ontology* into corresponding Rust and C# representations. The API for accessing it is also generated.

3.1.4 Interval Algebra

Reasoning about traffic situations requires a solution for handling the temporal aspect of it. From all the approaches discovered during literature review (see section 2.2) and presented in subsection 2.1.8, *Allen's Interval Algebra* constitutes the best fit. Therefore, it was chosen for the representation of time, time intervals and relations between them in the whole assessment system. The details about it can be found in already mentioned subsection 2.1.8. In addition, Figure 2.5 describes the basic relations between time intervals as defined by Allen [27].

3.1.5 Pattern Query Engine

The subsection 3.1.2 has already described a suitable ontology representation and similarly the subsection 3.1.4 has defined a reasonable time representation. The goal of this subsection is to bring these two aspects together while allowing comprehensive and efficient reasoning i.e. queries to *Knowledge Graph*. The implemented solution is based on *Description logic* (2.1.9) and inspired by the work of several researchers, namely Hulsen et al. [70], Jinhuan Wang and Baomin Li [67], Buechel et al. [65], Fuchs et al. [57] and Zhao et al. [49].

The reviewed work predominantly relies on existing reasoners designed for reasoning about OWL ontologies. Unfortunately, the reasoners are not compatible

with the data-oriented *Knowledge Graph* structure. Therefore, the developed solution employs a custom reasoning system named *Pattern Query Engine* which allows searching for patterns in *Knowledge Graph* within time and space. As mentioned before, the system utilizes *Description logic* but supports only a subset of it. The set of reasoning capabilities include: *ABox – Concept assertions*, *Role assertions* and *Individual (non)equality*, *TBox – Concept equivalence* but no *Concept inclusion*, *RBox – Role equivalence* but no *Role inclusion*. The engine also does not support constructors for *Concepts* and *Roles*. In contrast to *Description logic*, the engine supports temporal reasoning and utilizes *Allen's Interval Algebra* (subsection 3.1.4) for the representation of time.

Figure 3.8 presents an example of a pattern query. The left side of the figure shows a hypothetical state of *Knowledge Graph* with *Ego* node and four *Car* nodes connected by *isBehind* and *isInFront* relations. The query is defined as follows (using an illustrative pseudo code): *Node(typeOf(Ego)) with outgoing Relation(typeOf(isBehind)) to Node(typeOf(Car))*. After the execution of the query, a result in the form of a tree structure is returned (see Figure 3.8). Even though only two nodes and a relation between them were specified, the engine returned a tree structure with three nodes as there are two possible patterns to match. For the query execution, one also needs to provide a time interval within which the query evaluates. Because of the fact that *Knowledge Graph* has different states at different times, the query might be successful for some particular time interval while being unsuccessful for some other interval (i.e. returning an empty result). Moreover, due to the time interval relations defined by *Interval Algebra*, it is possible to combine queries in time e.g. *search for pattern 1 within interval (0,10) and if found search for pattern 2 within interval (10, 15)* is a perfectly valid query.

The worst case computational complexity of a pattern query is $O((n_1 + n_2...) + (r_1 + r_2...))$ where each n_x represents one node that is a part of the query and each r_x represents one relation which is also a part the query. The values of both n_x and r_x represent the number of records of node/relation indices of the corresponding types. Thanks to the separation of index record types, the actual number of records tends to hold low in comparison with the total number of index records. In addition, the underlying linear arrays make the lookup quite cache efficient which further speeds up the query execution.

All of these features combined form a robust solution for reasoning about traffic situations. Importantly, the whole solution has proven its capability to deliver results in real-time which is particularly important for the correct and timely operation of the assessment system (see chapter 5).

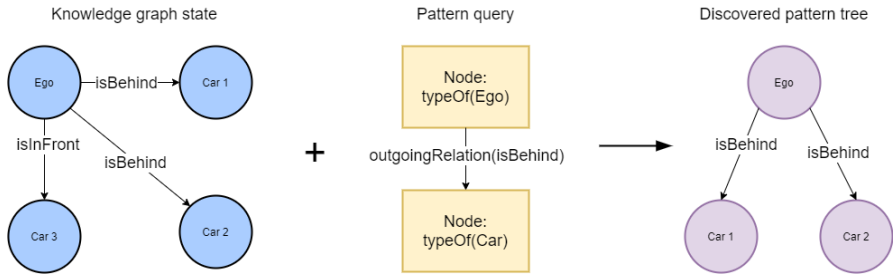


Figure 3.8: The execution flow of a query in *Pattern Query Engine*.

3.2 Assessment System Architecture

The core part of the solution is an assessment system application whose purpose is to obtain sufficient level of *Situation Awareness* (2.1.5) and to reason about various traffic situations. The system was designed and implemented as multi-agent (2.1.4) and built around *Knowledge Graph* and *Pattern Query Engine* concepts described in previous subsections (3.1.2 and 3.1.5). In addition, the solution was inspired by multi-agent systems presented by Doniec et al. [41], Nguyen [46], Gutierrez et al. [44], Weiss and Steger [52] and Sukthankar et al. [59].

The multi-agent system described in this section is commonly referred to as *secondary multi-agent system* as there also exist *primary multi-agent system* described later in subsection 3.3.1. The agents of *secondary multi-agent system* are often called *high-level agents* whereas the agents of *primary multi-agent system* are *low-level agents*. In fact, each multi-agent system maps to a different level of *Situation Awareness* model proposed by Endsley [19]. *Primary multi-agent system* deals with achieving the first level of *Situation Awareness* (Perception of elements in current situation) while *Secondary multi-agent system* gains the second level of *Situation Awareness* (Comprehension of current situation). The third level (Projection of future status) is not considered in this thesis.

3.2.1 Entity Component System

The assessment system architecture is based on *Entity Component System* architectural pattern (2.1.3) which enforces modularity and follows the composition over inheritance principle. Furthermore, this pattern is highly compatible with Data-oriented design principles (2.1.2). The foundation of the design was laid by Bosse et al. [42] and Danielsson and Bohlin [45] and subsequently further extended in this work. In addition, the final ECS implementation was influenced

by the work of Garcia and de Almeida Neris [43], Hodson and Millar [50] and Hall [54].

In the actual system, *Components* are represented as various custom defined structures of data stored in linear *Component storage* vectors. Intelligent agents map to *Entities* which, therefore, are commonly referred to as *Agent Entities*. The reasoning logic of each agent is encapsulated in *Systems* and hence can be easily swapped with a different solution. Moreover, the system introduces the notion of *Modules*. *Modules* are additional units containing source code which can be used to enhance the functionality of the assessment system. An example of a module could be a new reasoning engine or some data processing algorithms packed in a library. Each *Agent Entity* has one or more *Source Components* and exactly one *Sink Component*. The *Source Components* provide the *Agent Entity* with the data necessary for it reasoning process. On the contrary, *Sink Component* stores the result of the *Agent Entity*'s reasoning process. The whole ECS architecture is shown in Figure 3.9. The actual implementation of the provided *Agent Entities* is presented and discussed in section 3.4.

As you may have seen in Figure 3.9, individual *Component Storage* vectors are accessed through *Read handle*s and *Write handle*s. This is because of the reason that the vectors are designed to be thread-safe, lock-free (see subsection 2.1.12) and eventually consistent data structures. Their design was inspired by Lange et al. [74] but is to a large extent based on the solution of Gjengset et al. [76]. Both of the mentioned papers use hash maps as the underlying data structure. This work takes the eventually consistent double-buffered hash map designed by Gjengset et al. [76] and adapts the approach to make it work with linear vectors. We named the resulting data structure *Eventually Consistent Vector* or ECV.

The internal workings of ECV are illustrated in Figure 3.10. The data structure works with arbitrary number of *Read handle*s i.e. readers and one *Write Handle* i.e. writer. Multiple writers are possible if locking is introduced (unwanted and not necessary for the purposes of the assessment system). As mentioned in the previous paragraph, the data structure is essentially double-buffered i.e. exists in two copies. One of the copies serves for reads and the second one for writes as shown in the left third of Figure 3.10. Thus, the essential question is when is it safe to swap the pointers pointing to each of the copies in order to publish newly written data (the middle third of Figure 3.10). The work of Gjengset et al. [76] introduces local epoch counters for each of the readers. Every time a reader does an operation (a read) on the vector, it increments its local counter before it starts the operation and also after it finishes the operation. Then, when the

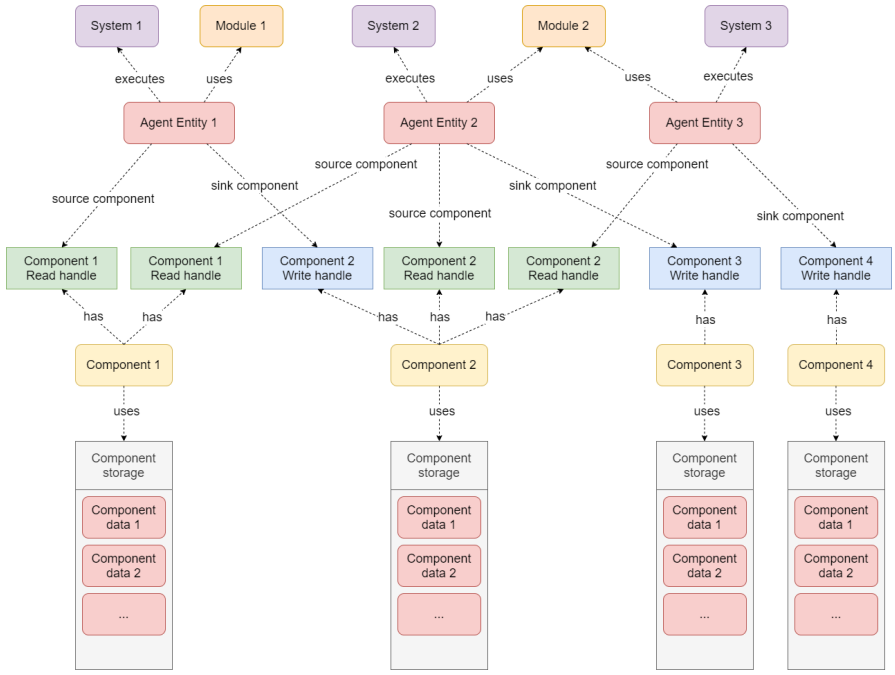


Figure 3.9: The schema of the ECS architectural pattern adapted for the multi-agent assessment system purposes.

writer (there is only one) wants to do the pointer swap, it first swaps the pointers and then waits until it sees all the epoch counters either having an even value (signifies inactive reader) or sees them increase by 1 (reader has finished its operation). Finally, to keep the data consistent after a swap, the writer reapplies the writes it did to the previous vector to its new vector (the right third of Figure 3.10).

3.2.2 Agent Platform

Agent Platform is a platform that utilizes the custom ECS solution described in subsection 3.2.1 and hosts all *Components* and their corresponding storage vectors, *Agent Entities*, *Systems* and *Modules*. Importantly, the platform takes care of work distribution between multiple threads and execution scheduling which helps to achieve real-time performance. Furthermore, it also provides an inter-

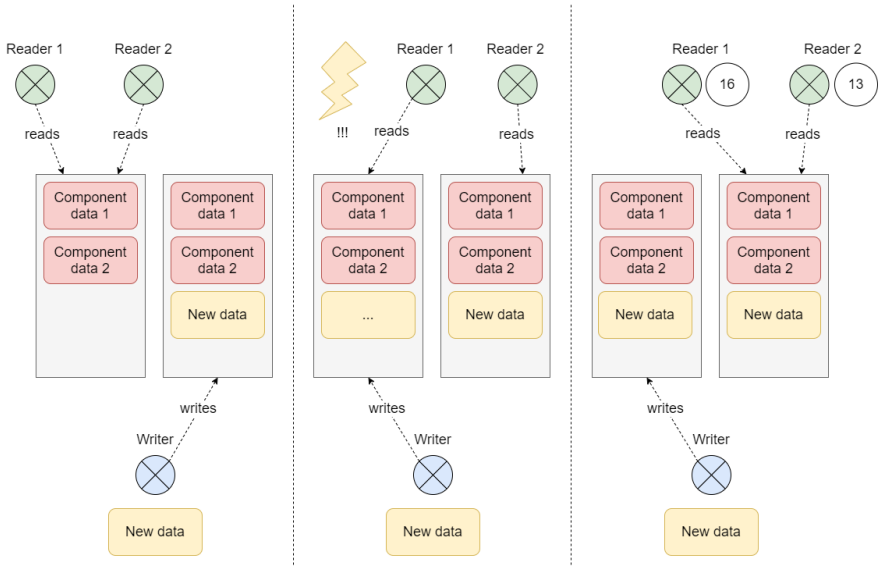


Figure 3.10: The internal workings of ECV demonstrated on three data structure states.

face for the driving simulator (see 1.2). The idea of real-time agent platform was introduced in the work of Krol and Nowakowski [66].

All *Component*s used by the multi-agent assessment system must be first explicitly registered in the platform. However, the platform also contains some *Implicit Components* which form the bridge between the system and the driving simulator. *Implicit Components* actually provide access to shared memory regions described in subsection 3.3.4. Similarly, each included *Agent Entity* must be registered beforehand too. In order to register an *Agent Entity*, its *Source Component*s, *Sink Component* and *System* must be specified. In addition, *Agent Entity* can be registered as one of the two available types. The types are described below:

1. **Periodic**: *Agent Entity* runs periodically after some set period of time elapses. The period needs to be provided while registering the entity.

2. **Observing**: *Agent Entity* runs after all the other *Agent Entities* it observes

produce new data. The observed *Agent Entities* need to be provided while registering the entity.

In a typical *Agent Platform* configuration, the first line of *Agent Entities* consists of *Periodic* entities while the latter entities keep waiting for data produced by other entities i.e. are of the *Observing* type. Therefore, *Observing* entities in fact have a data dependency on the observed entities i.e. form a *Dependency Graph* as described by Wang [77]. An example configuration including several *Agent Entities* is illustrated in Figure 3.11. In *Agent Platform*, each of the registered entities runs in its own thread and communicates with the platform and other entities through message passing. The messages are just simple notifications, the actual data is exchanged through *Components* and the underlying thread-safe and lock-free vector based storages. The various kinds of implemented *Agent Entities* are later described in section 3.4. Similarly to *Agent Entities*, individual *Modules* also need to get registered in the platform beforehand in order to become available.

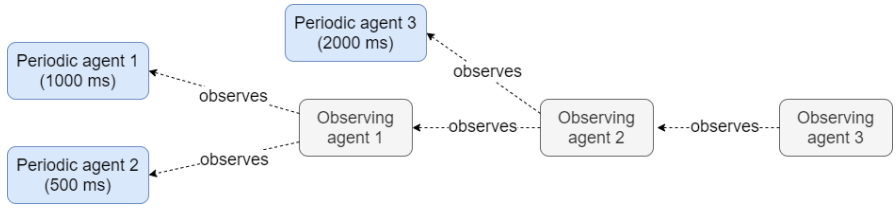


Figure 3.11: Example agent *Dependency graph* which combines various *Observing* and *Periodic* agents. The number of milliseconds below each *Periodic* agent name signifies the amount of elapsed time after which they get periodically triggered.

3.3 Simulator Interface

For the purpose of data extraction from the driving simulator, a suitable interface needs to be designed. The interface ideally has to be as generic as possible while supporting extraction of all the data required by the assessment system. Also, the sole pulling of data from the simulator is not a sufficient nor a complete solution. A bridge which connects the assessment system to the simulator interface needs to be built too.

3.3.1 Unity Interface

The simulator and all available simulated worlds (basically various driving lessons) were developed by Way AS on top of Unity real-time development platform (see subsection 2.1.11). Therefore, the required interface needs to utilize the platform as well while respecting its limitations and constraints. The goal of the simulator interface is to obtain *Situation Awareness* by constructing an appropriate *Knowledge Graph* representation of the current traffic situation. Then, the graph needs to be made available for the assessment system.

For the *Knowledge Graph* construction part a set of *low-level agents* so called *primary multi-agent system* (subsection 2.1.4) was developed and hosted inside of Unity (for more information see section 3.2). The whole multi-agent system is represented by a single simulated world entity while its individual agents run inside of *coroutines* provided by the platform. Each of the agents focuses on detection of a different phenomenon. More details about the individual implemented agents is provided in subsection 3.4.3.

In order to determine which simulated world elements of interest are part of the current traffic situation, the platform's collision engine was utilized. The *Ego* car was encapsulated in four collision triggers which bring various world elements into scope as they get closer, and remove them as they get further away. Moreover, multiple triggers allow *detection hysteresis* i.e. one trigger is marked as a trigger that detects the appearance of a simulated world element while another trigger deals with the removal of the same element. This helps to remove flickering of elements that are located close to trigger boundaries. However, the rule this solution enforces is that the exit trigger needs to fully encapsulate the corresponding enter trigger for the hysteresis to work correctly. The individual triggers are described below and also visualized in Figure 3.12:

1. **Outer trigger**: The outermost (red) trigger. Deals only with the removal of elements that are required to stay in scope the longest.

2. **Inner trigger**: The second biggest (violet) trigger. Usually utilized as the entry trigger while being coupled with *outer trigger*.

3. **Nearby trigger**: The middle sized (gray) trigger. Often paired with the previous trigger (*inner trigger*) while serving as the entry trigger.

4. **Immediate trigger**: A trigger (yellow) that exactly matches the shape of the *Ego* car. Most commonly used for detecting close interactions of *Ego* such as collisions with other cars or pedestrians.

One trigger pair forms data input/output for low-level agents of the secondary multi-agent system. In general, the triggers can be paired freely based on the data needs of each agent.

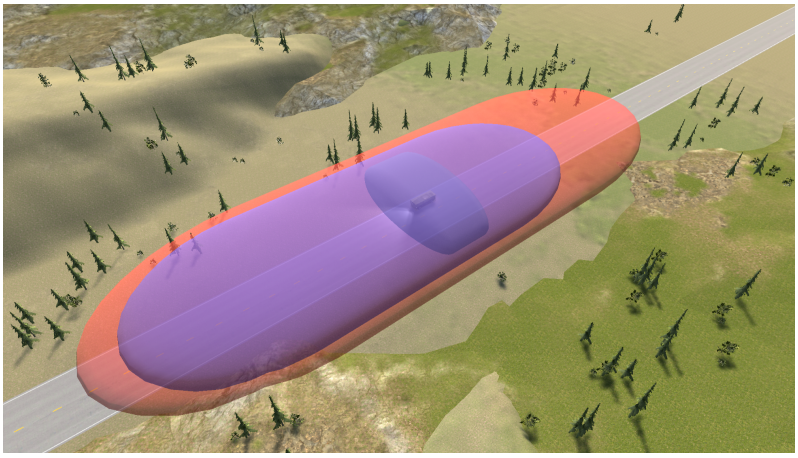


Figure 3.12: Four different collision triggers of *Ego* which enable the perception of simulated world elements.

3.3.2 Road Network

One of the primary elements of interest is the road network. Its accurate representation constitutes a complex problem in real world as well as in the simulated environment. For the simulated world representation purposes, a custom road network representation and appropriate *lane marking tool* were developed. The network representation was based on the solution of Su et al. [55] and also inspired byBuechel et al. [65], Hulsen et al. [70] and Zhao et al. [49].

The employed representation is based on *lanes* i.e. every possible segment of road network is represented solely as the lanes it contains. Figure 3.13 shows how the representation looks like for a two lane road as well as for a 3-way intersection. The intersection case is a bit more interesting as it contains lanes for

all possible passes a car can make through the intersection. Some of the lanes overlap, therefore, for a car it is possible to be in multiple lanes at the same time. The resolution of the correct lane is performed by agent/agents of *secondary multi-agent system*. Other road network segments are represented in a similar way. Example visualisations of already marked lanes from some of the simulated worlds are shown in Figure 3.14 and Figure 3.15.

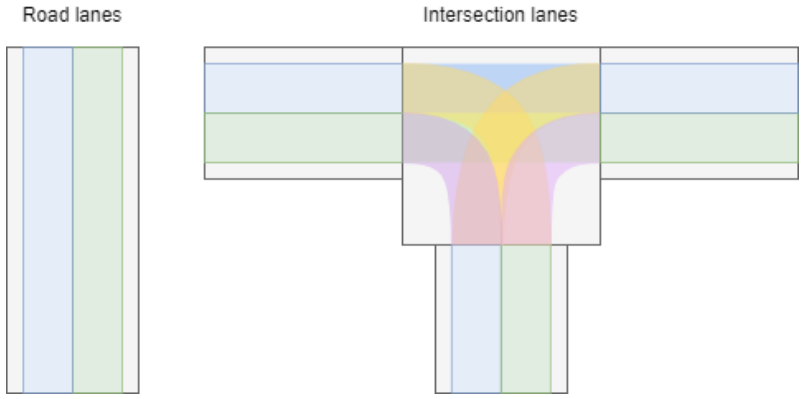


Figure 3.13: Lane-based representation of a road (left) and a 3-way intersection (right).

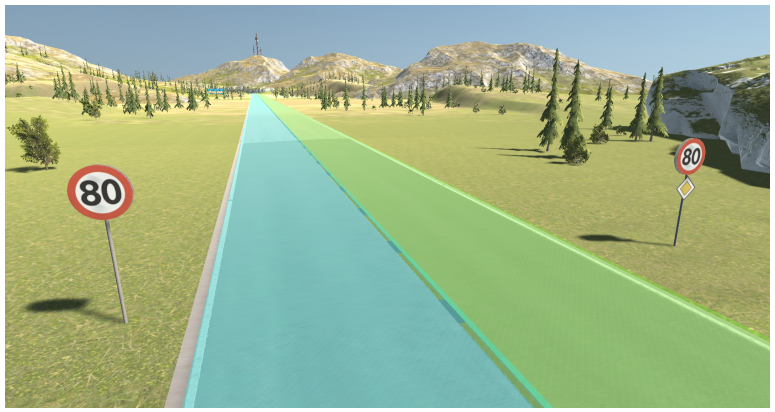


Figure 3.14: Visualisation of marked road lanes in one of the simulated worlds.

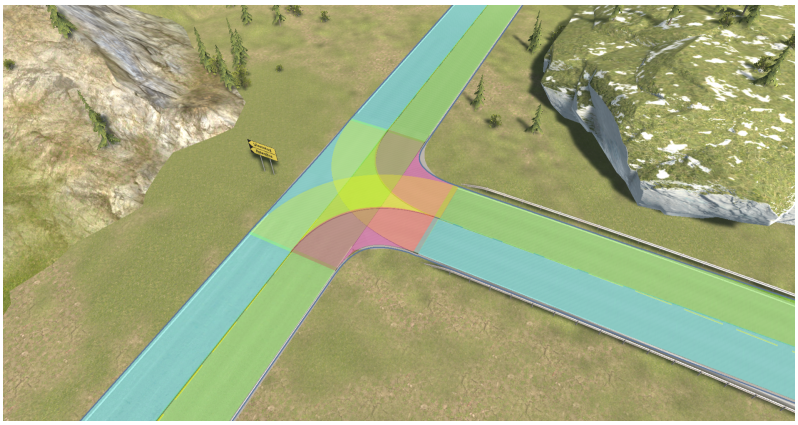


Figure 3.15: Visualisation of marked intersection lanes in one of the simulated worlds.

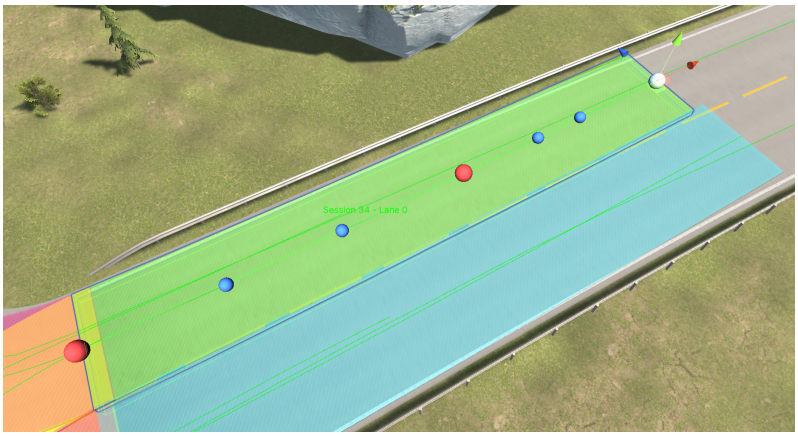


Figure 3.16: Visualisation of the *Lane marking tool* interface. The points that define the lane spline are marked red while the control points are marked blue.

In order to speed up the lane marking process, a custom *lane marking tool* was developed. The tool is spline-based i.e. the lane marking process essentially boils down to just laying out splines. The lane splines are defined by spline points and their shape can be further refined by manipulating their control points. The lane color, width or depth can be easily customized from the tool's interface.

The interface for spline manipulation is shown in Figure 3.16. After the marking is finished, the tool automatically generates geometry representing the marked lanes and turns it into active collision trigger zones. These zones are then used for the detection of *Ego* or other *Car* lanes.

3.3.3 Environment Tagging

Apart from lane marking, any object of the simulated world can be assigned some additional metadata in a manual *tagging* process. The metadata can then be utilized by secondary agents and propagated further to *Knowledge Graph*. The metadata is assigned in a form of *tag components* which are represented as C# scripts in Unity (2.1.11). The provided implementation supplies five basic *tag components* which are listed in Table 3.3.

Tag Component	Properties	Internal State
LaneTag	id, name, speed limit, parent road	-
RoadTag	id, name	-
IntersectionTag	id, name, type, size	cars at intersection
RoadSignTag	id, type, properties, parent lane	-
CrosswalkTag	id, name, parent lane	pedestrians on crosswalk

Table 3.3: A table listing all available *tag components*.

The existing *tag components* can be modified and also new *tags* can be added to enable annotation of a greater range of simulated world objects. Example visualisations of tagged crosswalks and intersections are shown in Figure 3.17 and Figure 3.18 respectively.

3.3.4 Shared Memory

The simulator software uses a shared memory region for the distribution of the global simulation state among all of its individual nodes (computers). The synchronization of this region between nodes is carried out over custom low-level Ethernet-based protocol. The global simulation state contains a lot of data which is also useful for the assessment system such as ego speed, direction or current gear. Therefore, the shared region forms the first contact point to interface with.

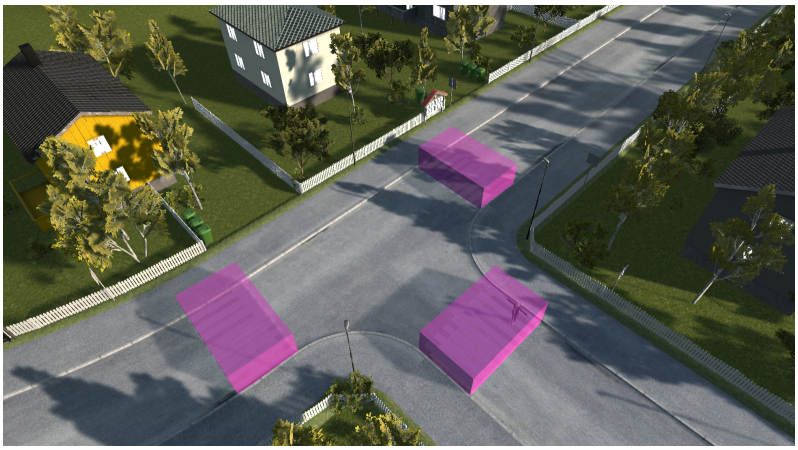


Figure 3.17: Visualisation of several tagged crosswalks.

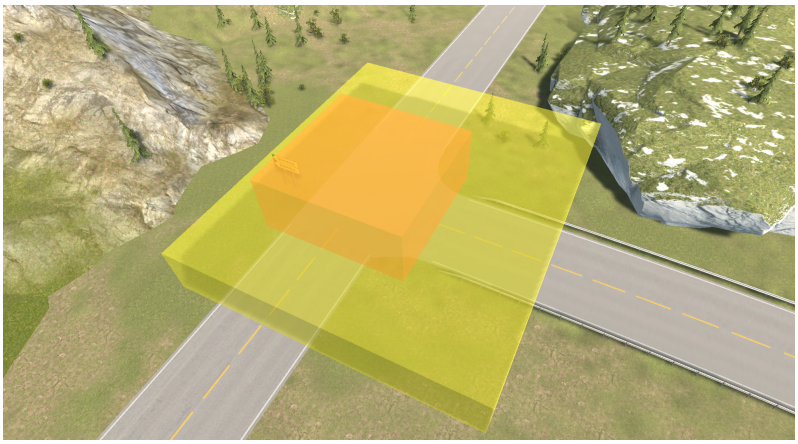


Figure 3.18: Visualisation of a tagged intersection.

The second crucial data to exchange is the *Knowledge Graph* instance so far considered isolated inside of the simulator software (i.e. inside of Unity platform, see subsection 2.1.11). Fortunately, the same shared memory region approach can also be utilized for storing and distributing the whole *Knowledge Graph* structure. This solution allows the simulator to maintain the same communication medium

with the additional capability of delivering all necessary data to the assessment system application as well. The application then simply reads the data from both shared memory regions. The situation is illustrated in Figure 3.19. The figure also shows the possibility of running the assessment system on a separate node (computer). This feature was not used even though it is supported. The assessment system runs on the master node.

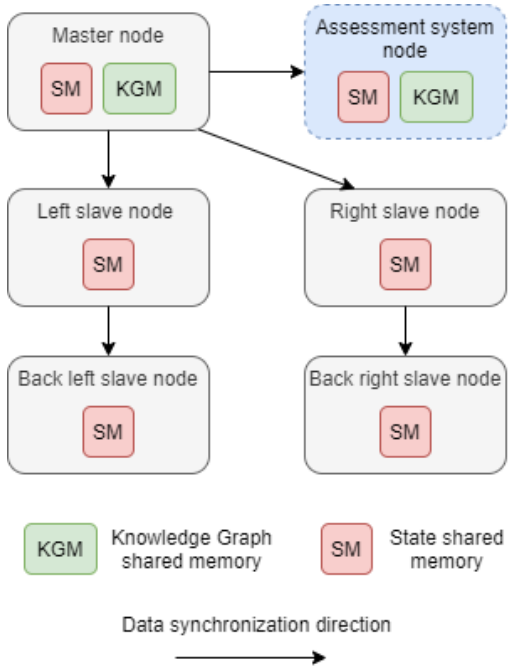


Figure 3.19: The distribution of shared memory regions over the different simulator nodes (computers).

3.4 Assessment System Agents

The assessment system implementation includes in total 18 intelligent agents for the purpose of system functionality demonstration and testing. *Primary multi-agent system* hosts 6 of the agents and *secondary multi-agent system* the remaining 12. The goal of both multi-agent systems (see section 3.2) and the corresponding agents is to provide assessment of five basic driving skills, namely:

the use of a *correct gear* (3.4.5), *speeding* (3.4.6), performing an *overtake* (3.4.7), *yielding to cars* (3.4.8) at traffic light controlled intersections and *yielding to pedestrians* (3.4.9) at crosswalks.

The assessment results produced by various agents are in real-time communicated to the driver through the simulator audio interface (car speakers). Apart from agents concerned with assessment, *secondary multi-agent system* also contains *feedback agents* which decide and schedule feedback to be given. The feedback is based on *text templates* or *segments* which are firstly augmented with produced assessment data and then converted to audio via cloud-based Text-To-Speech (for more information, see subsection 2.1.14) solution. In this work, free version of Google Cloud Text-To-Speech service was used.

3.4.1 Modules

As described in subsection 3.2.1, the assessment system allows inclusion of various modules in order to allow arbitrary extension of its capabilities or functionality. For the five proposed driving skill assessments (3.4), two modules were needed and hence implemented. The modules are listed below:

- **Fuzzy logic module**: A module which encapsulates and delivers generic fuzzy logic reasoning engine (see subsection 2.1.10). The module was utilized for the assessment of speeding (3.4.6).

- **Text-To-Speech module**: A module that allows conversion of text segments into spoken speech. Internally, the module makes requests to Google Cloud Text-To-Speech service, downloads the results and finally performs the audio playback. The module has been frequently utilized by most of the *feedback agents*.

3.4.2 Components

The basis for data exchange in *secondary multi-agent system* are *Component*s and their storages (3.2.1). Table 3.4 lists all defined and used *Component*s. The links between various *Component*s and individual agents are shown in Figure 3.21. The first two *Component*s listed in Table 3.4 are *Implicit Component*s (3.2.2).

Component	Data	Description
Knowledge Graph	*Knowledge Graph* instance	Provides reference to *Knowledge Graph* stored in the shared memory.
Ego Data	position, direction, speed, gear, throttle, brake, clutch...	Pulls data about *Ego* from state shared memory.
Incorrect Gear	gear state	Stores two basic states of wrong gearing.
Incorrect Gear Feedback	value	String value holding the given feedback text.
Speeding	speeding state, speed limit, ego speed	Stores speeding information as a basis for feedback.
Speeding Feedback	value	String value holding the given feedback text.
Lane Change	from lane, to lane	Stores data about the previous and new lane.
Turn Signal	turn signal state	Stores two basic states of turn signaling.
Overtake	timestamp, duration, turn signals	Stores overtake information as a basis for feedback.
Overtake Feedback	value	String value holding the given feedback text.
Car Yielding	timestamp, type	Stores yielding information as a basis for feedback.
Car Yielding Feedback	value	String value holding the given feedback text.
Pedestrian Yielding	timestamp, type	Stores yielding information as a basis for feedback.
Pedestrian Yielding Feedback	value	String value holding the given feedback text.

Table 3.4: A table listing all *Component*s used by agents of *secondary multi-agent system*.

3.4.3 Low-level Detection Agents

As mentioned in section 3.4, there are 6 low-level agents forming the *primary multi-agent system*. These agents are responsible for building *Knowledge Graph* inside of the simulator software (3.3.1). A diagram showing all of them can be

found in Figure 3.20. Moreover, their detailed description is provided in Table 3.5. In fact, the implemented agents can be informally divided into two groups: *active* and *passive*. *Active* agents run repeatedly during the whole time a simulated world element is in scope. On the other hand, *passive* agents run only when an element is detected or lost.

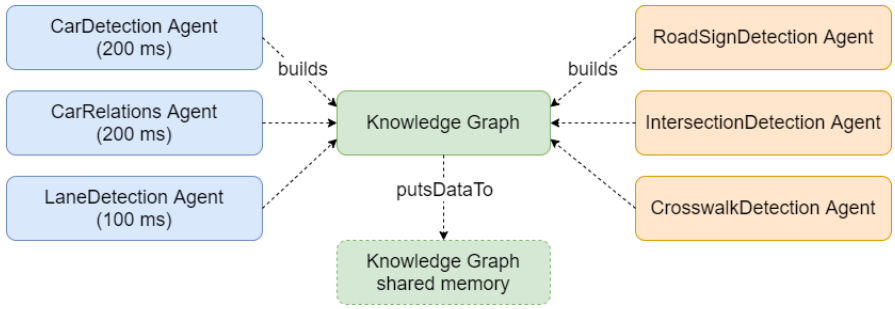


Figure 3.20: Low-level agents also known as *primary multi-agent system*. The blue agents are *active* agents while the orange agents are *passive*.

Agent Name	Entry/Exit Trigger	Description
Car Detection Agent	inner/outer	Detects cars nearby *Ego*
Car Relations Agent	-	Determines spatial relations to nearby cars detected by *Car Detection Agent*
Lane Detection Agent	immediate	Detects the current lane of *Ego*
Road Sign Detection Agent	near/inner	Detects road signs for the current lane
Intersection Detection Agent	immediate/near	Detects when *Ego* enters intersection
Crosswalk Detection Agent	immediate/near	Detects when *Ego* drives over crosswalk

Table 3.5: A table listing all agents of *primary multi-agent system*.

3.4.4 High-level Assessment Agents

All implemented high-level agents are shown in Figure 3.21 together with their corresponding *Component*s and *Module*s (for more information see subsection 3.2.1).

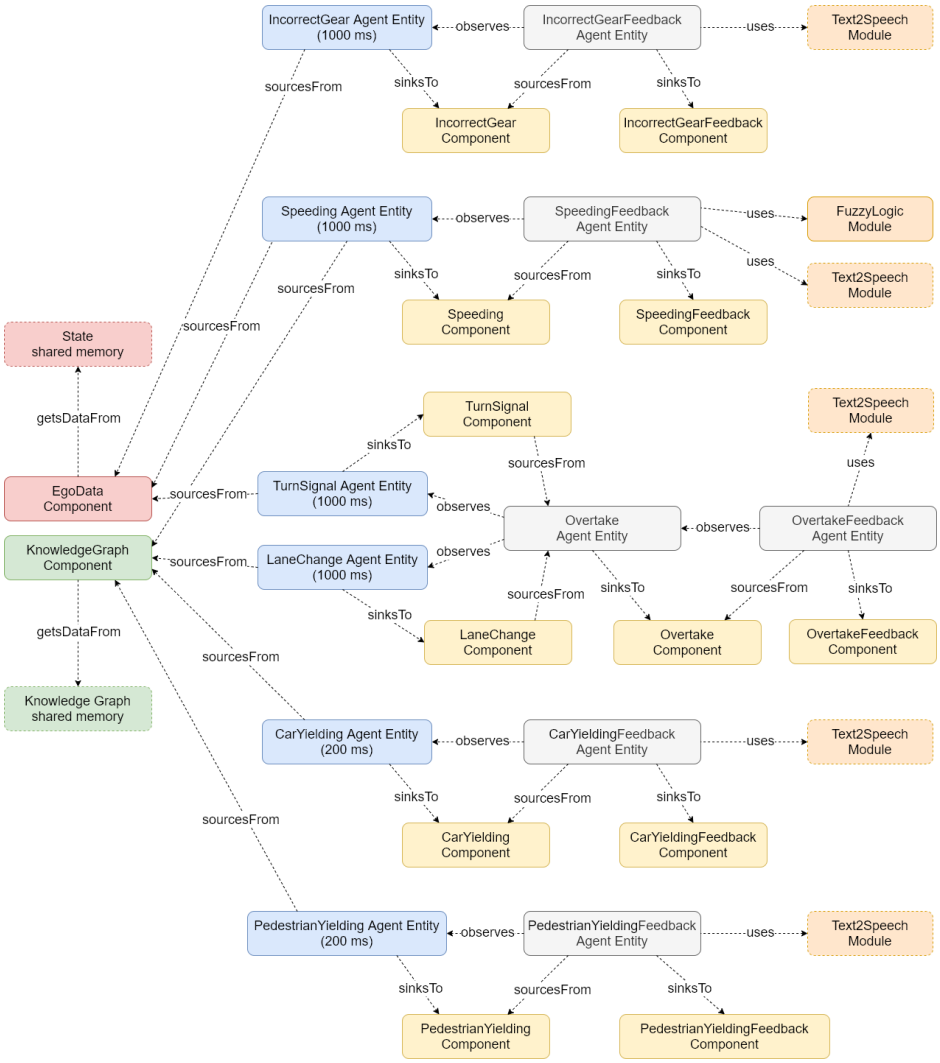


Figure 3.21: High-level agents also known as *secondary multi-agent system*.

For the sake of clarity, the description of high-level agents was split into several sections based on which driving skill assessment are the individual agents responsible for. In addition, each section contains a cropped version of Figure 3.21

showing only the relevant agents. The sections follow below and one by one describe approaches to assessments of five chosen driving skills.

3.4.5 Incorrect Gear

The first driving skill to assess is the correct use of gears. For this purpose, two agents were developed as shown in Table 3.6 and Figure 3.22. *Incorrect Gear Agent* reads data from ***Ego Data*** component and determines if the current RPMs of *Ego* engine fall in the correct range. Based on this information, the agent suggests shifting to either lower or higher gear. There exist two suitable RPM ranges: *normal range* from 1650 to 3400 RPMs and *acceleration range* from 2000 to 3800 RPMs. The active RPMs range is based on the current throttle amount which was normalized to 0-1 range. The threshold was set to 0.5.

Agent Name	Agent Type	Source Components	Sink Component
Incorrect Gear Agent	Periodic (1000 ms)	***Ego Data***	*Incorrect Gear*
Incorrect Gear Feedback Agent	Observing	*Incorrect Gear*	*Incorrect Gear Feedback*

Table 3.6: The list of all *Agent*s involved in incorrect gear assessment.

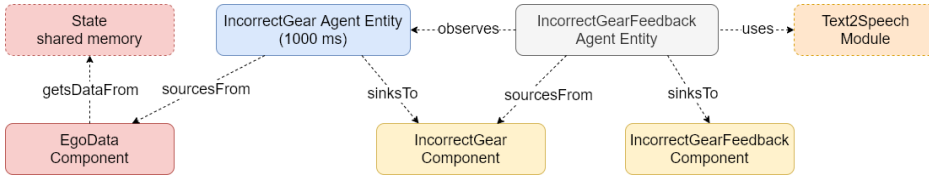


Figure 3.22: The illustration of all *Agent*s, *Component*s and *Module*s involved in incorrect gear assessment. The diagram was cropped out of Figure 3.21 which depicts all available high-level agents.

Incorrect Gear Feedback Agent takes care of delivering *Incorrect Gear Feedback* at the right time. The agent reads data from *Incorrect Gear* component and employs a simple rule for feedback scheduling. If there is 8 or more records (the number is configurable) of *Incorrect Gear* data with the same value in a row, then

feedback based on that value is enqueued. The agent randomly chooses from a set of predefined *text segments* which are then send to *Text-To-Speech module* in order to be played. The available *text segments* are listed in Table 3.7.

#	Assessment Result	Feedback Text
#1	*Too high gear*	You need to shift down!
#2	*Too high gear*	Mind your current gear!
#3	*Too high gear*	Your gear is too high!
#4	*Too high gear*	You should shift down!
#5	*Too high gear*	Shift down please!
#6	*Too low gear*	You need to shift up!
#7	*Too low gear*	Mind your current gear!
#8	*Too low gear*	Your gear is too low!
#9	*Too low gear*	You should shift up!
#10	*Too low gear*	Shift up please!

Table 3.7: The list of all *text segments* available for Text-To-Speech feedback provided by *Incorrect Gear Feedback Agent*.

3.4.6 Speeding

In the case of speeding assessment, fuzzy logic (see subsection 2.1.10) was utilized. The assessment logic is encapsulated in two agents (Table 3.8 and Figure 3.23): *Speeding Agent* and *Speeding Feedback Agent*. The former focuses on fuzzy logic reasoning whereas the latter schedules and delivers feedback. *Speeding Agent* reads from ***Ego Data*** in order to elicit information about the current speed and acceleration of *Ego*. In addition, the agent also executes two pattern queries (see subsection 3.1.5) which extract data from **Knowledge Graph** component.

Agent Name	Agent Type	Source Components	Sink Component
Speeding Agent	Periodic (1000 ms)	***Ego Data***, ***Knowledge Graph***	*Speeding*
Speeding Feedback Agent	Observing	*Speeding*	*Speeding Feedback*

Table 3.8: The list of all *Agent*s involved in speeding assessment.

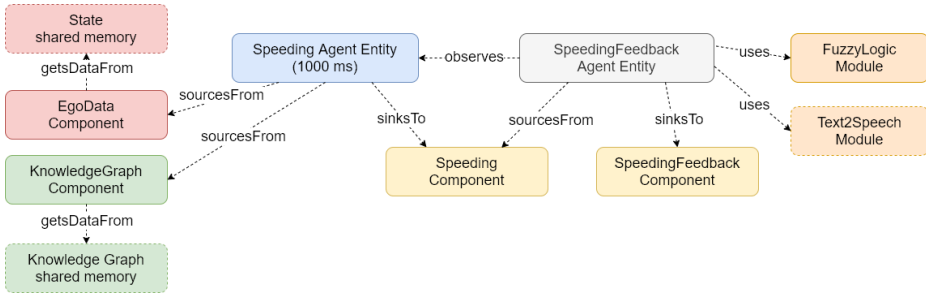


Figure 3.23: The illustration of all *Agent*s, *Component*s and *Module*s involved in speeding assessment. The diagram was cropped out of Figure 3.21 which depicts all available high-level agents.

The first pattern query: *Node(typeOf(Ego)) with outgoing Relation(typeOf(isIn)) to Node(typeOf(Lane))* determines *Ego*'s current lane while the second query: *Node(typeOf(SpeedLimitSign)) with outgoing Relation(typeOf(belongsTo)) to Node(typeOf(Lane))* probes if there is any *SpeedLimitSign* belonging to the lane *Ego* is currently in. Then all obtained information is fed to the *Fuzzy Logic* module for reasoning. The result of *Speeding Agent* calculations is stored in *Speeding* component.

The difference between *Ego* speed and the active speed limit is represented by *Over Speed Limit* fuzzy input variable. Furthermore, the acceleration of *Ego* comes in as *Acceleration* fuzzy variable. Their corresponding fuzzy sets are listed in Tables 3.9 and 3.10. The output is captured in *Speeding* fuzzy output variable as shown in Table 3.12. In addition, fuzzy rules used during the reasoning process are described in Table 3.11. The final defuzzification is not executed all the way i.e. the output value is still fuzzy which is desired for the purpose of speeding assessment.

Fuzzy Set	**Membership Function**	**Range** (km/h)
Ok	Reverse grade	(0, 20)
Slightly	Triangular	(0, 15, 20)
Considerably	Triangular	(5, 20, 35)
Severely	Grade	(30, 100)

Table 3.9: Fuzzy sets defined on *Over Speed Limit* fuzzy input variable.

Fuzzy Set	**Membership Function**	**Range (m/s^2)**
Decelerating a lot	Reverse grade	(-10, -2)
Decelerating	Triangular	(-4, -2, 0)
Constant	Triangular	(-2, 0, 2)
Accelerating	Triangular	(0, 2, 4)
Accelerating a lot	Grade	(2, 10)

Table 3.10: Fuzzy sets defined on *Acceleration* fuzzy input variable.

Rule	**If (AND)**	**Then**
#1	*Over Speed Limit* is Ok *Acceleration* is Decelerating	*Speeding* is Ok
#2	*Over Speed Limit* is Slightly *Acceleration* is Decelerating	*Speeding* is Ok
#3	*Over Speed Limit* is Considerably *Acceleration* is Decelerating	*Speeding* is Ok
#4	*Over Speed Limit* is Ok *Acceleration* is Decelerating a lot	*Speeding* is Ok
#5	*Over Speed Limit* is Slightly *Acceleration* is Decelerating a lot	*Speeding* is Ok
#6	*Over Speed Limit* is Considerably *Acceleration* is Decelerating a lot	*Speeding* is Ok
#7	*Over Speed Limit* is Ok *Acceleration* is Accelerating	*Speeding* is Ok
#8	*Over Speed Limit* is Slightly *Acceleration* is Accelerating	*Speeding* is Considerably Above
#9	*Over Speed Limit* is Considerably *Acceleration* is Accelerating	*Speeding* is Severe Speeding
#10	*Over Speed Limit* is Severely *Acceleration* is Accelerating	*Speeding* is Severe Speeding
#11	*Over Speed Limit* is Ok *Acceleration* is Constant	*Speeding* is Ok
#12	*Over Speed Limit* is Slightly *Acceleration* is Constant	*Speeding* is Slightly Above
#13	*Over Speed Limit* is Considerably *Acceleration* is Constant	*Speeding* is Considerably Above
#14	*Over Speed Limit* is Severely *Acceleration* is Constant	*Speeding* is Severe Speeding

Table 3.11: Fuzzy rules employed while reasoning about *Speeding*.

Fuzzy Set	Membership Function	Range
Ok	Reverse grade	(0, 5)
Slightly above	Triangular	(5, 10, 15)
Considerably above	Triangular	(15, 20, 25)
Severe speeding	Grade	(25, 100)

Table 3.12: Fuzzy sets defined on *Speeding* fuzzy output variable.

Speeding Feedback Agent works in a similar manner like *Incorrect Gear Feedback Agent* does (subsection 3.4.5). If *Speeding* component contains 8 or more new records about speeding (the number is configurable), the agent schedules feedback to play. The agent also utilizes a set of several *text segments* which are being sent to *Text-To-Speech module* for reading (see Table 3.13). The feedback is also stored in corresponding *Speeding Feedback* component.

#	Assessment Result	Feedback Text
#1	*Slightly above speed limit*	Mind your speed!
#2	*Slightly above speed limit*	You are slightly above the speed limit!
#3	*Slightly above speed limit*	Hey! Mind the speed limit.
#4	*Considerably above speed limit*	You are speeding now!
#5	*Considerably above speed limit*	You should lower your speed!
#6	*Considerably above speed limit*	Please lower your speed!
#7	*Severe speeding*	This is unacceptable speeding!
#8	*Severe speeding*	Lower your speed immediately!
#9	*Severe speeding*	Hey! You are severely speeding!

Table 3.13: The list of all *text segments* available for Text-To-Speech feedback provided by *Speeding Feedback Agent*.

3.4.7 Overtake

The assessment of overtakes required four agents (Table 3.14), namely: *Lane Change Agent*, *Turn Signal Agent*, *Overtake Agent* and *Overtake Feedback Agent*. *Lane Change Agent* periodically reads ***Knowledge Graph*** component via the following pattern query: *Node(typeOf(Ego)) with outgoing Relation(typeOf(isIn)) to Node(typeOf(Lane))* and detects all lane changes that happen. The lane changes are saved as *Lane Change* component records. *Turn Signal Agent* reads

data about turn signaling from ***Ego Data*** component and produces *Turn Signal* records every time an active turn signal is detected. Whenever a lane change is detected *observing Overtake Agent* also runs.

Overtake Agent traces *Lane Change* component records to see if there were two subsequent lane changes back and forth between two distinct lanes. If these lane changes are found, the agent tries to look up relative locations of other cars from ***Knowledge Graph*** component. If a pattern comprising of a sequence of *isBehind*, *isOnLeft*, *isInFront* relations to the same *Car* in this temporal order is found, an overtake is registered and saved as *Overtake* component record.

Agent Name	Agent Type	Source Components	Sink Component
Lane Change Agent	Periodic (1000 ms)	***Knowledge Graph***	*Lane Change*
Turn Signal Agent	Periodic (1000 ms)	***Ego Data***	*Turn Signal*
Overtake Agent	Observing	***Knowledge Graph***, *Lane Change*, *Turn Signal*	*Overtake*
Overtake Feedback Agent	Observing	*Overtake*	*Overtake Feedback*

Table 3.14: The list of all agents involved in overtake assessment.

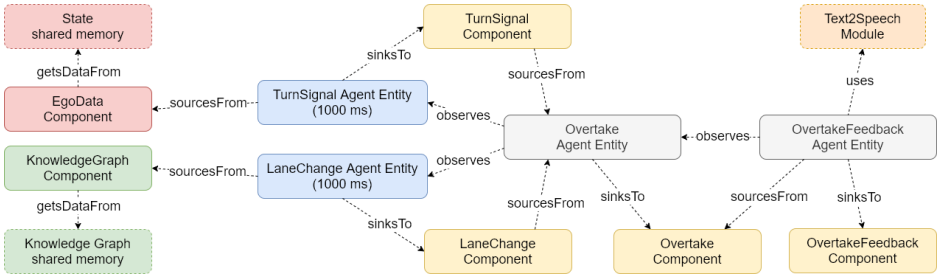


Figure 3.24: The illustration of all *Agent*s, *Component*s and *Module*s involved in overtake assessment. The diagram was cropped out of Figure 3.21 which depicts all available high-level agents.

The three corresponding pattern queries look as follows: *Node(typeOf(Ego)) with outgoing Relation(typeOf(isBehind)) to Node(typeOf(Car))*, *Node(typeOf(Ego)) with outgoing Relation(typeOf(isOnLeft)) to Node(typeOf(Car))*, *Node(typeOf(Ego)) with outgoing Relation(typeOf(isInFront)) to Node(typeOf(Car))*. Finally, turn signaling records in *Turn Signal* component storage are searched to see if turn signals were used properly during the detected overtake.

In this case, the job of *Overtake Feedback Agent* is fairly simple. Each time a new *Overtake* component record is saved, the agent runs and produces feedback. Similarly to previous agents, it utilizes *Text-To-Speech module* and the feedback is based on predefined *text segments*. The *text segments* available to *Overtake Feedback Agent* are listed in Table 3.15. The given feedback records are saved as *Overtake Feedback* component.

#	Assessment Result	Feedback Text
#1	*No turn signals*	You performed an overtake without turn signals!
#2	*No turn signals*	Remember turn signals while overtaking!
#3	*No turn signals*	You did not signal while overtaking!
#4	*Only left turn signal*	Nice overtake but you forgot to blink right!
#5	*Only left turn signal*	You did not use right turn signal while overtaking!
#6	*Only left turn signal*	Hey! Don't forget to signal left and then right during an overtake!
#7	*Only right turn signal*	Nice overtake but you forgot to blink left!
#8	*Only right turn signal*	You did not left turn signal while overtaking!
#9	*Only right turn signal*	Hey! Don't forget to signal left and then right during an overtake!
#10	*Both turn signals*	Perfect! Flawless overtake with both turn signals!
#11	*Both turn signals*	Great overtake! Nothing to complain about!
#12	*Both turn signals*	Good! This was a very well done overtake!

Table 3.15: The list of all *text segments* available for Text-To-Speech feedback provided by *Overtake Feedback Agent*.

3.4.8 Car Yielding

As yielding to other cars is a complex subject to assess, the scope of the problem was narrowed down to yielding at traffic light controlled intersections only. The assessment is performed by two agents (Table 3.16 and Figure 3.25): *Car Yielding Agent* and *Car Yielding Feedback Agent*. The former simply searches the ***Knowledge Graph*** component for patterns containing relations between

Ego's current lane and other lanes with other *Car*s that *Ego* is supposed to yield to (two pattern queries in total: *Node(typeOf(Ego)) with outgoing Relation(typeOf(isIn)) to Node(typeOf(Lane))* and also *Node(typeOf(Lane)) with outgoing Relation(typeOf(yieldsToCarIn)) to Node(typeOf(Lane))*). Additionaly, the ***Knowledge Graph*** component is probed for a pattern indicating that *Ego* is currently entering an intersection (*Node(typeOf(Ego)) with outgoing Relation(typeOf(isAt)) to Node(typeOf(Intersection))*).

If appropriate patterns are found, yielding violation is saved as *Car Yielding* component record. *Car Yielding Feedback Agent* fires each time there is a new car yielding violation. Similarly to other feedback agents, it delivers the feedback as spoken speech derived from *text segments* (see Table 3.17). The feedback agent produces *Car Yielding Feedback* records.

Agent Name	Agent Type	Source Components	Sink Component
Car Yielding Agent	Periodic (200 ms)	***Knowledge Graph***, *Turn Signal*	*Car Yielding*
Car Yielding Feedback Agent	Observing	*Car Yielding*	*Car Yielding Feedback*

Table 3.16: The list of all agents involved in yielding to other cars assessment.

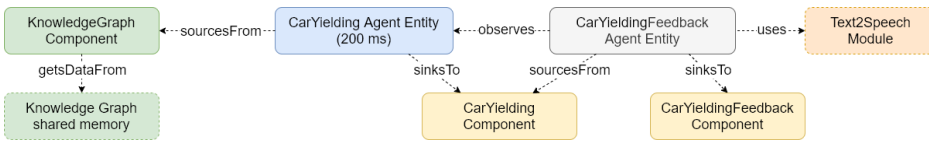


Figure 3.25: The illustration of all *Agent*s, *Component*s and *Module*s involved in yielding to other cars assessment. The diagram was cropped out of Figure 3.21 which depicts all available high-level agents.

#	Assessment Result	Feedback Text
#1	*Going on red light*	You passed an intersection on a red light!
#2	*Going on red light*	Hey! Going on red light is strictly forbidden!
#3	*Going on red light*	Pay attention to traffic lights! The light was red!

Table 3.17: The list of all *text segments* available for Text-To-Speech feedback provided by *Car Yielding Feedback Agent*.

3.4.9 Pedestrian Yielding

Pedestrian yielding assessment is nearly identical to car yielding assessment described in subsection 3.4.8. Practically, the only difference is the pattern the first agent searches for. In this case, *Pedestrian Yielding Agent* looks for a pattern in which *Ego* has *passesOver* relation to *Crosswalk* while the crosswalk is still occupied by pedestrians (*Node(typeOf(Ego)) with outgoing Relation(typeOf(passesOver)) to Node(typeOf(Crosswalk))*).

Agent Name	Agent Type	Source Components	Sink Component
Pedestrian Yielding Agent	Periodic (200 ms)	***Knowledge Graph***	*Pedestrian Yielding*
Pedestrian Yielding Feedback Agent	Observing	*Pedestrian Yielding*	*Pedestrian Yielding Feedback*

Table 3.18: The list of all agents involved in yielding to pedestrians assessment.

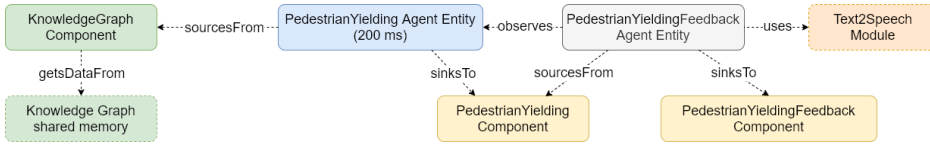


Figure 3.26: The illustration of all *Agent*s, *Component*s and *Module*s involved in yielding to pedestrians assessment. The diagram was cropped out of Figure 3.21 which depicts all available high-level agents.

If this situation is detected, *Pedestrian Yielding Feedback Agent* automatically triggers feedback. Further details and the utilized component types are described

in Table 3.18 and Figure 3.26. The available feedback *text segments* are listed in Table 3.19.

#	**Assessment Result**	**Feedback Text**
#1	*Unsafe crosswalk pass*	You drove over a crosswalk while people were still crossing!
#2	*Unsafe crosswalk pass*	Hey! You almost hit people crossing the street!
#3	*Unsafe crosswalk pass*	Pay attention to pedestrians on crosswalks! You have just put them in danger!

Table 3.19: The list of all *text segments* available for Text-To-Speech feedback provided by *Pedestrian Yielding Feedback Agent*.

Chapter 4

Experiments and Results

In this chapter, two experiments conducted to verify the correctness of the designed and implemented solution are described. Firstly, the chapter details the employed experimental plan and experimental setup. Then, the achieved results are presented accordingly. The experiments were performed during *Evaluation* phase of *Design Science Research* methodology (see section 1.4).

4.1 Experimental Plan

In order to evaluate the assessment system under conditions closest to a real-life scenario, application-grounded evaluation strategy was employed. Application-grounded evaluation relies on conducting experiments with domain experts within a real application task [78]. Hence, two driving lessons in simulated environment originally developed by Way As (section 1.2), namely: *Overtake lesson* and *Traffic light controlled intersections lesson* were used and modified for the evaluation purposes.

The simulator interface described in section 3.3 was embedded in both of the lessons to enable communication with the rest of the assessment system. Road network lanes marking (subsection 3.3.2) and tagging of relevant simulated world elements (subsection 3.3.3) also took place. All agents described in section 3.4 were utilized in order to test the system under load and also provide rich assessment capabilities. Therefore, the assessment of five driving skills was available: *the use of correct gears*, *speeding*, *overtakes*, *yielding at traffic light controlled intersections* and *yielding for pedestrians at crosswalks*. The results of each skill assessment were provided as spoken speech supplied in real-time by *feedback agents* (3.4.4) during the actual driving session.

Each lesson constitutes one experiment, therefore two experiments were performed in total. For the evaluation of both experiments, adequate domain experts i.e. driving teachers were chosen. However, the set of available driving teachers was limited to 8 employees of Way As only, as the intellectual property of the company cannot be disclosed to other driving schools. No actual driving school students were involved in the experiments.

The execution process was the same for both experiments. Each experiment follows the same evaluation plan to ensure consistency and allow production of comparable results. A domain expert drives through both of the simulator lessons (one at a time) while following the standard lesson outline. As shown in Table 4.1 each lesson/experiment is split up into three 5 minutes long parts each of which has a different purpose. The purposes and corresponding instructions for domain experts are further detailed in Table 4.1. Before the start of every evaluation session, domain experts were made familiar with the plan and also given all the necessary instructions. Furthermore, during each lesson/experiment, they were guided by the same evaluation supervisor who was making sure they did not misunderstand any instructions, providing potential guidance, and also keeping track of the elapsed time. During each lesson session with a domain expert a log of the assessment system operation was produced. All logs are available for download in Appendix C (6.2).

Part	**Length**	**Instructions**	**Purpose**
#1	5 min	Drive as correctly as possible while following traffic rules to the highest possible extent.	Tests if the system produces only adequate *positive* or *no feedback* at all. *(no false positives)*
#2	5 min	Perform the most common mistakes that students often do.	Tests if the system correctly delivers appropriate *negative feedback*. *(no false negatives)*
#3	5 min	Try to confuse the system and intentionally force it to produce incorrect assessment.	To stress the system and discover potential flaws or weak spots.

Table 4.1: The evaluation plan used for both of the performed experiments. *False positives* and *false negatives* mentioned in **Purpose** column are described from the correct *negative feedback* perspective.

After both lesson/experiment sessions were finished, aggregated evaluation was performed. At first, a semi-structured interview took place. The goal of the interview is to elicit as much of qualitative data from domain experts as possible while also making them reflect on their recent experience. Primarily, the data collection focus was on domain expert opinions about correctness, timing and completeness of the produced feedback. However, questions concerning several other aspects such as their initial expectations, support of such system future development or opinions about the usefulness of this educational approach are also included. The actual interview questions can be found in Appendix A (6.2).

Finally, the domain experts were confronted with a questionnaire. The purpose of the questionnaire is to accompany the qualitative data collected during interviews with additional quantitative data. Therefore, the questionnaire consists of Likert scale (Joshi et al. [79]) based questions only. Their range was from 1 (the worst) to 7 (the best). The data are then used to perform statistical analysis of the domain experts evaluation. The questionnaire template can be found in Appendix B (6.2), however, the used questions are also presented in subsection 4.3.1.

4.2 Experimental Setup

The experiments, interviews and filling in of questionnaires all took place at Way As offices in Trondheim (section 1.2). For the evaluation performed by one domain expert, 1.5 hours of time was allocated beforehand. However, there was an option for potential extension if necessary. At first, each domain expert drove through *Overtake lesson* and then through *Traffic light controlled intersections lesson* (each of the lessons took 15 minutes as described in Table 4.1). Unfortunately, some of the domain experts experienced problems with motion sickness caused by the driving simulator. Therefore, short pauses needed to be introduced in between each lesson part (individual lesson parts are described in Table 4.1). After both lessons were finished, an interview took place in a calm and separate room hosting two participants only (the evaluation supervisor and domain expert). Afterwards, domain experts were asked to fill in a questionnaire.

4.3 Experimental Results

This section presents results obtained from experiments. For the sake of clarity, the section is divided in two subsections. First subsection presents the analysis of quantitative data collected from questionnaires whereas the second subsection examines qualitative data gathered during interviews. Both subsections are also

further divided based on the content of used questions. The questions naturally form two groups. The first group aggregates questions concerned about *general opinions* while the second group of questions is focused on the *assessment of individual driving skills*.

4.3.1 Quantitative Data

For the quantitative analysis of questionnaire data, two main approaches were taken. In the first approach, the total number of answers to each individual step of Likert scale [79] for each question was aggregated and plotted as a bar chart. The chart for *general opinions* group of questions is shown in Figure 4.1 while the chart for *assessment of individual driving skills* group is presented in Figure 4.2. The Figures essentially illustrate how the distribution of answers looks like. The questions that the Figures refer to are listed in Table 4.4.

In the second approach, the *mean* of all answers to each question was calculated which we also commonly refer to as *super domain expert*. Then, *standard deviation* was calculated too in order to determine how far the individual answers are from each other as well as how far they are from *super domain expert*. Similarly, the results for *general opinions* group of questions are shown in Table 4.2 and Figure 4.3 while the results for *assessment of individual driving skills* group are presented in Table 4.3 and Figure 4.4. The questions that the data refer to are again listed in Table 4.4.

#	**Mean** (1-7)	**Standard deviation**
Question 1	3.5	0.926
Question 2	3.375	1.188
Question 3	4	1.512
Question 4	3.625	1.188
Question 5	4.625	1.408
Question 6	2.375	1.061
Question 7	5.75	0.463
Question 8	5.875	0.354
Question 9	4.25	1.488

Table 4.2: *Mean* and *standard deviation* of all answers to each question from *general opinions* group. The results are also visualised in Figure 4.3.

Finally, it is important to note that the analyzed data set contains only 8 records as there were only 8 domain experts available. In order to derive some reliable

conclusions from the data set, one would need to collect greater amount of records.

#	**Mean** (1-7)	**Standard deviation**
Question 1	3.5	1.4142
Question 2	4.375	1.061
Question 3	3.375	1.303
Question 4	3.75	0.707
Question 5	4.125	1.246

Table 4.3: *Mean* and *standard deviation* of all answers to each question from *assessment of individual driving skills* group. The results are also visualised in Figure 4.4.

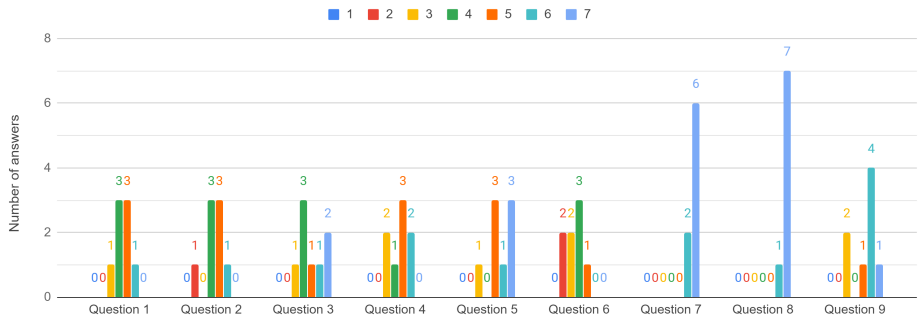


Figure 4.1: The distribution of answers for *general opinions* group of questions.

4.3.2 Qualitative Data

For the qualitative analysis of data collected during interviews, *opinion counting* method was used [9]. Essentially, all transcribed interviews were reviewed in order to discover opinions and comments that are either interesting on their own or shared between at least two domain experts. Similarly to quantitative data analysis (subsection 4.3.1), the data was split up into two groups: *general opinions* and *assessment of individual driving skills*. Furthermore, during the *opinion counting* process, we identified 9 topics that domain experts commonly talked about. Thus, *general opinions* group was further divided into 10 subsections (the

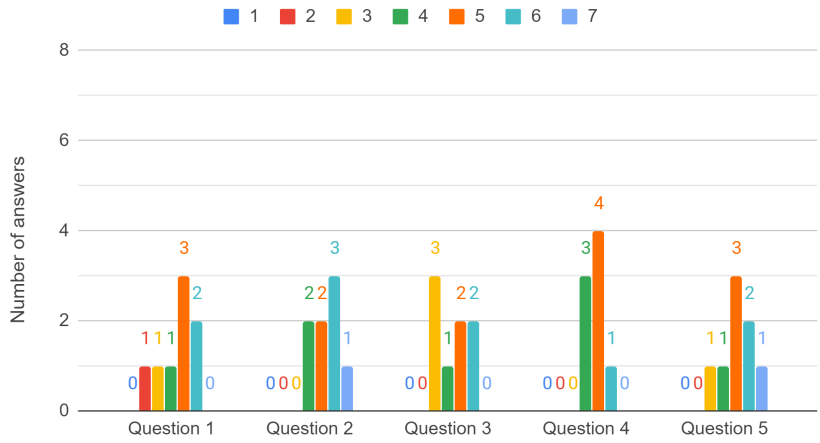


Figure 4.2: The distribution of answers for *assessment of individual driving skills* group of questions.

last subsection covers the interesting comments mentioned during interviews). At first, the results for *general opinions* group are presented below. The fractions in parentheses represent the portion of domain experts that shared the same opinion.

Additional value for driving education

Half of the domain experts (4/8) thinks that this kind of a system will provide entertainment and fun experience to students. Smaller portion (2/8) sees the system as a valuable addition to driving students education and would rate the experience as positive.

Prior expectations

More than one third (3/8) of experts would rate the performance of the system either the same as expected or slightly worse than expected. On the other hand, smaller portion (2/8) would rate the system performance as better than expected.

Feedback content

When it comes to feedback content, majority of the domain experts (5/8) claimed that it was easy to understand the mistake made based on the feedback. Half of

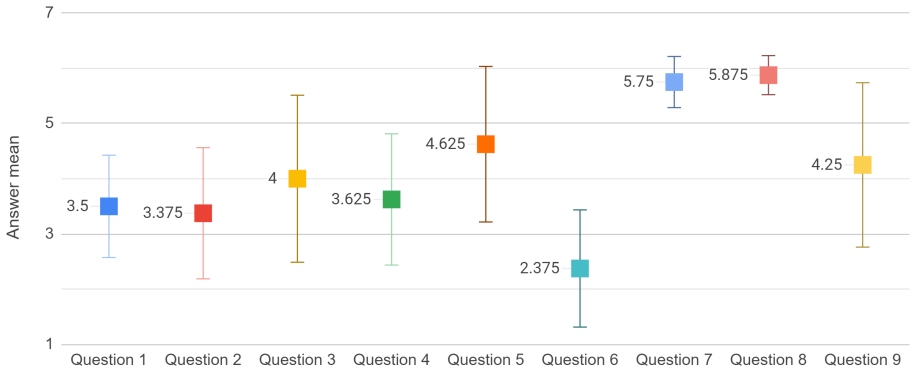


Figure 4.3: *Mean* and *standard deviation* of all answers to each question from *general opinions* group. The whiskers show the range of mean +/- standard deviation.

them (4/8) either stated that it was clear how to avoid the mistake in the future, that the voice was clear and understandable, that there should be feedback which only provides information and does not talk about mistakes, or that there should be more positive and praising feedback in general. Half of the experts (4/8) also mentioned and asked about the possibility of having a conversation with the feedback system. Minority of experts (3/8) would welcome more information about how to avoid the mistake in the future. Finally, two people (2/8) mentioned that there should be more feedback given overall.

Feedback timing

For the majority of the domain experts (5/8), the timing of the feedback was good. Nearly half of them (3/8) mentioned that the priority of feedback was sometimes good and sometimes not ideal (depending on the situation). Two of the experts (2/8) also mentioned that it might be even counter productive or distracting if feedback is given at a bad time.

Feedback type

Most of the experts (5/8) think that audio/spoken speech is a good way of providing feedback. However, minority (3/8) of them also finds visualisations in the simulated environment helpful. Moreover, another minority of experts (3/8)

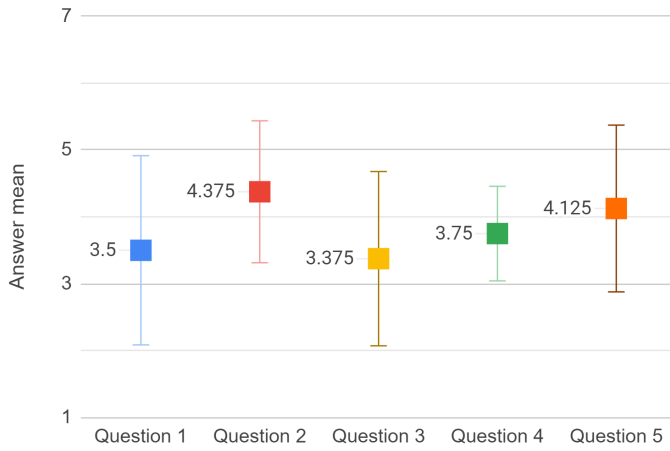


Figure 4.4: *Mean* and *standard deviation* of all answers to each question from *assessment of individual driving skills* group. The whiskers show the range of mean +/- standard deviation.

thinks that visualisations in the environment are helpful but only for new inexperienced students, not later in the educational process. Two of the domain experts (2/8) would like to see the system taking over student's control in a case of emergency.

Students' thinking process and self-reflection

All of the domain experts (8/8) think that the system needs to be able to focus on students' decisions and enforce thinking processes i.e. make students self-reflect and reason about their behaviour. Two of the experts (2/8) noted that the system needs to encourage students to develop understanding of their decision consequences.

Situation awareness and risk assessment

Most of the people (6/8) mentioned that the system needs to increase the scope of its situation awareness (see subsection 2.1.5) i.e. consider more things while providing assessment information. Two people (2/8) also noted that a proper risk assessment was missing. Furthermore, two people (2/8) think that aggressive way of driving should be taken into account.

Student skill level and feedback adaptation

Majority of the domain experts (5/8) would welcome a feedback that progressively adapts to the current student skill level.

Human factor of driving teachers

Half of the people (4/8) considers the provided feedback close to how a real teacher would give feedback to students. On the other hand, a small portion of experts (2/8) thinks that the system is missing the human aspect of a real teacher a bit.

Interesting points and remarks

Even though often mentioned just by a single domain expert, some of the notes constitute a lot of good ideas. The following enumeration is a collection of the most interesting ones:

- Mirror look assessments were missing.
- Sometimes traffic rules need to be relaxed.
- Assessment of unclear situations is needed but hard to do.
- Students might not take the education in simulator seriously.
- There should exist a potential feedback aggregation.
- Briefing and debriefing before and after a lesson.

Secondly, the results for *assessment of individual driving skills* group are presented below:

The use of correct gears

Half of the domain experts (4/8) complained about being forced to shift down to the first gear while driving.

Speeding

Some of the experts (2/8) wanted to have too low speed assessment as well. Furthermore, two of them (2/8) also think that speeding assessment should in general allow some small margin (as real speedometers usually overestimate a bit).

Overtakes

Nearly half of the people (3/8) would welcome safe distance to other cars assessment during overtakes. Also, two domain experts (2/8) mentioned that the duration of turn signalling while overtaking is an important aspect to assess as well.

Yielding at traffic light controlled intersection

More than one third of domain experts (3/8) thinks that correct positioning at intersections or turn signaling at intersections are really important things to include in the assessment. A bit smaller portion of them (2/8) claimed having problems with incorrect assessment if traffic lights changed to red while they were driving through an intersection. Some of the experts (2/8) also pointed out that the correct distance to an intersection should be considered.

Yielding for pedestrians at crosswalks

Two of the experts (2/8) consider this assessment to be really good. However, another two of them (2/8) pointed out that the speed while approaching crosswalks needs to be taken into account as well.

#/@	**Question**	**Likert Scale** (1-7)
#1	How would you rate your general experience with the assessment system?	Very dissatisfied <-> Very satisfied
#2	How informative was the feedback overall?	Very non-informative <-> Very informative
#3	How good was the timing of the feedback?	Completely off <-> Very good
#4	How good was the feedback in terms of accuracy?	Completely off <-> Very good
#5	Was it easy to understand the voice?	Very hard and unclear <-> Very easy and clear
#6	How "human" was the feedback?	Very "non-human" and quite far from a real teacher behaviour <-> Very "human" and close to a real driving teacher
#7	Do you believe that a system like this has potential and can be a helpful tool for simulator-based driving education?	Strongly disagree <-> Strongly agree
#8	Do you think that a system like this should be worked on and developed further?	Strongly disagree <-> Strongly agree
#9	Were there any issues with the system during your evaluation?	A lot <-> None
@1	How good was the assessment of correct/incorrect gear?	Very bad <-> Very good
@2	How good was the assessment of speeding?	Very bad <-> Very good
@3	How good was the assessment of overtakes?	Very bad <-> Very good
@4	How good was the assessment of yielding at traffic light controlled intersections?	Very bad <-> Very good
@5	How good was the assessment of yielding to pedestrians at crosswalks?	Very bad <-> Very good

Table 4.4: The questions that were part of the evaluation questionnaire (see section 4.1). The table is divided in two groups. The first group (#) represents *general opinions* questions whereas the second group (@) gathers *assessment of individual driving skills* questions.

Chapter 5

Evaluation and Conclusion

This chapter covers the evaluation of artifacts produced during *Suggestion* and *Development* phases of *Design Science Research* methodology (see section 1.4). Furthermore, the results obtained from experiments (chapter 4) are also considered and evaluated accordingly. Then, the discussion of discovered limitations is presented. Finally, the opportunities for potential future work are outlined. The content of this chapter maps to *Evaluation* and *Conclusion* phases of *Design Science Research*.

5.1 Discussion

In this section, we revisit hypothesis, goal and research questions originally presented in section 1.3 in order to relate them to the actual research outcomes and the experiment results described in section 4.3. At first, we address the research questions one at a time. All of the questions were answered successfully.

Research question 1 *Which data-oriented design principles can be utilized for the design of the assessment system?*

As described in subsection 3.1.2, for *Knowledge Graph* data structure the main utilized Data-oriented design principles are: *DBMS-like memory layout, Linear and continuous data structures, Data packing and sorting* and *Hot/cold splitting*. In addition, *Component storage*s which constitute *Agent Platform*'s (3.2.2) one and only data storage option make use of *DBMS-like memory layout* and *Linear and continuous data structures* principles too. The design principles are in detail explained in subsection 2.1.2. Moreover, the assessment system utilizes Entity Component System architectural pattern (2.1.3) which builds on top of

Components-based architecture principle. Finally, *Observing* agents running on *Agent Platform* employ *Existential processing* design principle.

Research question 2 *How can the system reason about traffic situations?*

In order to enable reasoning about complex traffic situations, dynamic ontology-driven approach was used (see section 3.1). The ontology serves as an abstraction of simulated world and allows capturing of not only of the current world state but also of all the previous states the world has ever had. The ontology is implemented as *Knowledge Graph* (3.1.2) which allows fast processing and traversing at runtime. On the top of *Knowledge Graph*, *Pattern Query Engine* (3.1.5) was built and utilized as a reasoning engine. *Pattern Query Engine* implements Description logic based reasoning combined with temporal reasoning capabilities provided by *Allen's Interval Algebra* (2.1.8).

Research question 3 *How can the system utilize concurrency on multi-core systems?*

The concurrency was utilized by multi-agent systems on per agent basis. Each agent of *secondary multi-agent system* (3.2 and 3.2.2) runs in its own thread and therefore in parallel with all other agents of the same multi-agent system. This solution guarantees that potential slow execution of one agent's reasoning process cannot negatively influence the performance of the other agents. In addition, parallel execution offers significant speedups and is one of the main principles allowing the assessment system to deliver results in real-time. The agents which are part of *primary multi-agent system* (3.2 and 3.3.1) are running on Unity development platform (2.1.11) inside of *coroutines*, therefore their utilization of concurrency is being handled internally by the platform.

Research question 4 *How can the system interface with a driving simulator in order to extract the necessary data?*

For simulator data extraction and proper interfacing with the assessment system, several techniques were designed and applied. At first, the simulated world elements and road network needed to be sufficiently annotated with metadata. Therefore, *lane marking* and *environment tagging* processes described in subsection 3.3.2 and subsection 3.3.3 were employed. Then, *primary multi-agent system* was used for *Knowledge Graph* construction and its later exposure to the assessment system. Finally, shared memory regions were utilized to allow sharing and exchange of data between different processes i.e. the simulator and the assessment system. Further details are provided in subsection 3.3.4. This collection of solutions makes interfacing with the simulator possible.

Research question 5 *Is it possible for such system to deliver driving skills assessment in real-time?*

In order to achieve real-time driving skill assessment delivery, two main things were necessary: timely evaluation of the skill assessment and an appropriate way of presenting the result in real-time. The first part was ensured by the assessment system design (3.2), as it is designed to work and provide results in real-time. For the second part, *feedback agents* (subsection 3.4.4) and *Text-To-Speech* (subsection 2.1.14) module (subsection 3.4.1) were utilized. Their purpose is to deliver feedback to a driver at the correct moment, potentially immediately (real-time) if desired.

Goal *Design and develop a data-oriented multi-agent assessment system which is capable of operating in real-time while providing the targeted driving skills evaluation.*

In order to draw a reliable conclusion about the achievement of the main goal of this research, more evidence was needed. The additional evidence was provided by experiments conducted with domain experts as described in chapter 4. When it comes to the collected quantitative data (subsection 4.3.1), the Tables 4.2 and 4.3 seem to contain data of the highest informational value (Table 4.4 lists the questions the charts refer to).

When it comes to *individual opinions* group, answers to *Question 1* (general experience) and *Question 2* (informative aspect of the feedback) were quite similar. The mean of all answers (also known as *super domain expert*) was close to the middle value of Likert scale {3.5} and standard deviation was close to {1}. Mean of answers to *Question 3* (timing of feedback) scored a little bit higher {4} but there also was higher standard deviation {1.5} i.e. the domain experts' opinions about the timing were more dissimilar. The situation around *Question 4* (accuracy of feedback) is also very similar to *Question 1* and *Question 2*. The mean of answers to *Question 5* (how easy it was to understand the voice) was the third highest in this question group {4.625} while the standard deviation was still holding around {1.4}, therefore we can claim that that domain experts tend to consider the feedback easy to understand. In contrast, the mean of *Question 6* (how human the assessment system feels) scored the lowest in this question group {2.375} with standard deviation around {1}, therefore the domain experts in general do not consider the system human enough. However, answers to *Question 7* and *Question 8* scored the highest mean {5.75 and 5.875} and lowest standard deviation in this group sitting below {0.5}. Based on these results, we can assume that the domain experts see the potential of this work and support the future development of such assessment system. The last question *Question 9*

(issues with the system) answers had mean of {4.25} and standard deviation close to {1.5}. However, it is important to note that the majority of domain experts considers missing functionality an issue, therefore the informational value of this question result is really low.

Assessment of individual driving skills question group contains even less diversity than *individual opinions* group. Most of the question means were around {3.75} with standard deviation close to {1.2}. There are just two notable exceptions. *Question 2* (speeding) scored the highest mean {4.375} which means that the assessment of *speeding* is considered to be the best among all other assessments. The second exception appeared in connection with *Question 4* (yielding at traffic light controlled intersections), which had the lowest standard deviation {0.7} i.e. the domain experts agreed the most on answers to this question.

The size of the data set used for quantitative data analysis made the derivation of statistically significant conclusions very difficult. Therefore, during qualitative data analysis (subsection 4.3.2) we put more strictness and emphasis on the number of people that share the same opinion during interviews. For that reason, we set a limit based on which are the discovered opinions considered either significant or not. The limit was set to *at least half or more of the domain experts*. The data is presented in descending order starting from the opinion that was mentioned the most.

Most of the domain experts (5/8) consider audio feedback a good way of providing feedback which supports the decision made in favor of Text-To-Speech (2.1.14) feedback system. The same amount of domain experts (5/8) also thinks that feedback was delivered at appropriate moments. This information highlights the timely evaluation of individual assessments as well as the correct feedback timing. Half of the domain experts (4/8) thinks that the feedback provided clear information about how to avoid the described mistake in the future and also considers the feedback content to be clear and understandable in general. This evidence emphasizes the correct assessment result and its appropriate presentation. Half of the domain experts (4/8) also believe the assessment system and provided feedback makes the assessment system valuable for driving education. This information to some extent confirms the results obtained during quantitative analysis (*Question 7 and 8*). On the other hand, half of the domain experts (4/8) also thinks that the system provides feedback feedback in a similar way like a real teacher would give feedback to students which contradicts the results of quantitative analysis (*Question 6*).

The qualitative data analysis uncovered some patterns that we can to a large

extent rely on while concluding about the main goal of this research. The quantitative data are harder to trust and therefore were utilized only as a data source describing tendencies rather than providing final conclusions. However, based on data from both data sources, the system seems to be considered a good foundation with a lot of potential for future improvement. To sum up, we see the main goal as achieved while providing a further discussion about limitations and future work in sections 5.2 and 5.3.

Hypothesis *Data-oriented multi-agent system for driving skills assessment in a simulated environment can be designed to run in real-time while supporting all functionality required for complete and timely driving skills evaluation.*

As all of the research questions were answered and also the main goal of this research is considered accomplished, therefore, we conclude that it is possible to design and implement data-oriented multi-agent system for driving skills assessment which will run in real-time while providing the desired complete and timely driving skills evaluation. To sum up, the hypothesis is considered to be well-supported by evidence and therefore assumed to hold true.

5.2 Limitations

As briefly mentioned in section 5.1, this section presents the limitations of the provided solution and the conducted research in general. When it comes to *general limitations*, one of the main hinders of a proper evaluation of this work (especially in a quantitative way) was the size of the obtained data set. Only 8 domain experts participated in the experiments (the reason for it is mentioned in section 4.1). Therefore, in order to derive more reliable conclusions, there needs to be more testing performed with more domain experts involved. Furthermore, there is a possibility of conducting experiments with driving education students as well to make the data collection process not limited to domain experts only. Additionally, only two driving lessons (out of 12 available) were modified and used for evaluation. More lessons would enable domain experts to experience the assessment system in a wider variety of traffic situations which will allow forming of better opinions about how well the individual assessments actually work. Finally, the system does not support the assessment of the third level of situation awareness (see subsection 2.1.5) i.e. is not able to predict future outcomes of traffic situations.

When it comes to limitations related to the *system design and implementation*, there are several identified limitations pertaining *Knowledge Graph* (3.1.2), *Pattern Query Engine* (3.1.5), *Agent platform* (3.2.2), *Unity interface* (3.3.1) and

also the implemented intelligent agents (3.4). *Knowledge Graph* has never been a subject to any kind of benchmarks for testing its real performance and efficiency. The fixed capacity of *Knowledge Graph* was necessary to make it compatible with shared memory regions (subsection 3.3.4) but might be a disadvantage in general for other use cases. The fixed capacity should be at least optional. During the implementation it also became apparent that the most common lookup which is performed on *Knowledge Graph* is iterating through timestamps. Hence, it might be worth moving timestamps to their own index structure which should speed up their frequent traversals.

Pattern Query Engine (subsection 3.1.5) suffers from incomplete support of *Description logic* and quite high computational complexity (as also described in subsection 3.1.5). In addition, the engine would benefit from its own query language as right now the queries are described directly in source code by creating instances of pattern structures. The use of markers in *Knowledge Graph* turned out to be sometimes problematic for *Pattern Query Engine*. As lookups performed on *Knowledge Graph* always start from the newest possible record, there might be issues with looking up elements in intervals during which the element changed state (i.e. was removed or updated). The lookup will always return the first found record even though in the rest of the interval the element might have been in a completely different state (for more information see subsection 3.1.5).

The *Component Storage*s of *Agent platform* are assumed to be fast linear data structures (as described in subsection 3.2.2), however, no benchmarking was performed. Moreover, agents running on the platform have their reasoning logic encapsulated in *System*s but still implemented in Rust (2.1.13) source code. There is a potential of abstracting the definitions of their logic into a higher level representation which would make it easier to create and modify.

The designed and implemented *Unity interface* (3.3) is so far compatible only with the simulator developed at Way As (1.1), however, general application of this work would require a more generic interface. The *lane marking tool* (3.3.2) and the corresponding lane marking process can easily get tedious and take a long time. As the simulator already contains an internal road network representation, there is a possibility of automatic generation of the marked lanes.

The implemented low-level and high-level intelligent agents (see subsection 3.4.3 and subsection 3.4.4) can be improved especially in terms of the scope of their detection or assessment. Also, in order to cover a wider variety of possible driving skill assessments, new agents should be developed as the current assessment capabilities are limited.

As we have low trust in quantitative data (4.3.1) obtained from experiments, we probe only the collected qualitative data (4.3.2) for limitations. Therefore based on the that data, all (8/8) of the domain experts think that the system needs to be able to focus more on students' decisions and enforce their thinking processes. Additionally, most of the domain experts (6/8) mentioned that the system needs to increase the scope of its situation awareness. Moreover, half of the domain experts (4/8) thinks either that there should exist feedback which only provides information and does not talk about mistakes or that there should be more positive and praising feedback in general. Finally, half of the experts (4/8) asked about the possibility of having a conversation with the assessment system. All the mentioned concepts or improvements are currently not available and therefore considered limitations of the system.

5.3 Future Work

The potential future work is closely tied to the limitations discussed in section 5.2. Basically, any of the proposed limitations constitute an opportunity for extensive future work. In addition, **Interesting points and remarks** group from subsection 4.3.2 lists some proposals for future work made by domain experts themselves.

Chapter 6

Publications

This chapter presents additional work that has been or will be published in a form of research papers.

6.1 General Research Paper

There is currently an ongoing work on a research paper which will reflect the work presented in this thesis.

6.2 Explanining Traffic Situations – Architecture of a Virtual Driving Instructor

The foundation of traffic situation ontology (3.1.1) and the approach to agent scheduling (3.2.2) presented in this thesis have served as a contribution to a research paper about Intelligent Tutoring Systems. The paper titled *Explanining Traffic Situations – Architecture of a Virtual Driving Instructor* was published and presented at 16th International Conference, ITS 2020 which took place at Athens, Greece, June 8–12, 2020.

Explaining Traffic Situations – Architecture of a Virtual Driving Instructor

Martin K. H. Sandberg[1], Johannes Rehm[1,2(✉)], Matej Mnoucek[1], Irina Reshodko[2], and Odd Erik Gundersen[1]

[1] Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway
{johannes.rehm,odderik}@ntnu.no,
martin.hoel.sandberg@gmail.com
[2] Way AS, Trondheim, Norway
irina@way.no

Abstract. Intelligent tutoring systems become more and more common in assisting human learners. Distinct advantages of intelligent tutoring systems are personalized teaching tailored to each student, on-demand availability not depending on working hour regulations and standardized evaluation not subjective to the experience and biases of human individuals. A virtual driving instructor that supports driver training in a virtual world could conduct on-demand personalized teaching and standardized evaluation. We propose an architectural design of a virtual driving instructor system that can comprehend and explain complex traffic situations. The architecture is based on a multi-agent system capable of reasoning about traffic situations and explaining them at an arbitrary level of detail in real-time. The agents process real-time data to produce instances of concepts and relations in an ever-evolving knowledge graph. The concepts and relations are defined in a traffic situation ontology. Finally, we demonstrate the process of reasoning and generating explanations on an overtake scenario.

Keywords: Virtual driving instructor · Intelligent Tutoring System · Situation awareness · Multi-agent system · Ontology · First order logic · Explanations

1 Introduction and Related Work

The field of Intelligent Tutoring System (ITS) has matured since the conception of the idea in 1970s [20], and ITS implementations have been used in various domains of life, such as crisis management [15] and vehicle driving training [2]. The main goal of ITS is to support effective learning and reduce workload of

This research was funded by Way AS and the Norwegian Research Council through the TRANSPORT program under the grant agreement 296640.

V. Kumar and C. Troussas (Eds.): ITS 2020, LNCS 12149, pp. 115–124, 2020.
https://doi.org/10.1007/978-3-030-49663-0_15

human teachers. In order to do that, an ITS must contain all the concepts, rules and decision-making approaches necessary for the domain awareness – in other words, it has to have an adequate *domain model*. It is also crucial to have a model of the cognitive state of the student and the learning tendencies – that is a *student model*. Finally, an ITS has to employ a *tutoring model* in order to choose the right form and timing of teaching feedback [20]. A very important additional requirement for an ITS is its explainability – it should be clear why the system made a decision. The sophistication of the domain comprehension necessary for effective teaching in most cases requires the use of an artificial intelligence (AI) system. According to Gunning and Aha, explainable AI is "AI systems that can explain their rationale to a human user, characterise their strengths and weaknesses, and convey an understanding of how they will behave in the future" [9]. Particularly, following correct traffic rules and the ability to produce explanations requires an adequate awareness about the situation [13]. Situation awareness (SAW) is obtained by perceiving all relevant elements in the environment, understanding their meaning and predicting their future state [5]. An ITS should be able to explain to the student how he or she handled a certain traffic situation, therefore it needs situation awareness and explainability.

One of the most widely used methods in AI, deep neural networks [8], have achieved tremendous results during the last decade. However, neural networks are inherently black box systems, and do not maintain an explicit awareness of a situation. Although there are attempts to make certain aspects of these systems explainable, such as deep explanations [7] and model induction [11], complete and interpretable explanations are still problematic. This is mainly due to the vast amount of operations performed in a deep neural network. The explainability requirement for ITS thus calls for the use of interpretable-by-design AI approaches. Ontologies that describe explicit relations between relevant concepts have been widely used to represent situations conceptually [15]. The multi-agent paradigm together with rule-based reasoning on an ontology has proven to be highly suitable for representing and reasoning with traffic situations [4].

Our work presents an architectural design of a virtual driving instructor (VDI) system, which uses the data from a virtual reality (VR) driving simulator in order to provide comprehensive integrated teaching assistance both online (during the driving lesson) and offline (as an after-lesson debrief). The virtual driving instructor should be able to reason about complex situations, i.e. situations that comprise of several sub-situations. Driving on public roads requires to perform multiple tasks in parallel, such as staying in the lane and maintaining a safe distance to the cars in front. Thus, the VDI system must be able to continuously assess several sub-situations at the same time. The explanation data which this system produces has to be as complete as possible for the traffic domain, and thus allow for a feedback of any level of detail.

Buechel *et al.* [3] propose an ontological framework for reasoning about various traffic scenarios which can be easily generalised to different traffic rule contexts. Their work focuses on regulation-aware decision-making for autonomous cars but lacks details and temporal aspects necessary for explanation purposes

in our proposed ITS. A traffic domain ontology was presented by Zhao *et al.* [22], and was tested to solve the yielding problem in narrow and uncontrolled intersections.

Zamora *et al.* [21] introduce a rule-based multi-agent architecture for an intelligent ADAS system assisting a driver in an urban environment. The main goal of the work by Zamora *et al.* is to recognise and display a warning about a potentially dangerous situation which the driver is not aware of, so it has no need for complex back in time reasoning or detailed explanations. This proposal has later been implemented and experimentally validated in [17]. Both works are based on the multi-agent approach described in [10]. The blackboard-based CarCOACH system developed by Arroyo *et al.* [1] uses a multi-agent architecture to assess scenarios such as hard braking. The CarCOACH system focuses on detecting and gently coaching the driver through immediately dangerous behaviour such as speeding while turning, and does not address more complicated scenarios or rule compliance. Sukthankar *et al.* [18] proposed and validated a multi-agent system with arbitrated voting for automated vehicle decision-making on a highway. As a vote-based system, it is not designed to provide a detailed explanation of its decisions. Weevers *et al.* [19] present a high-level multi-agent architecture of a virtual driving instructor for a commercial driving simulator. It can recognise and evaluate speed control and intersection handling, and is capable of reasoning if the student has performed certain driving tasks correctly with respect to the situation. However, the details of the architecture are not disclosed.

It is worth noting that an ontology-based multi-agent architecture is not the only way to approach the SAW problem. Raptis *et al.* [16] develop an attentiveness assessment system which tracks the hand gestures of a driver on the steering wheel using an SVM-based method to estimate an attention score. Many more approaches [14] exist for recognising and evaluating driving manoeuvres and behaviour.

The rest of the paper is structured as follows: In Sect. 2, we present our proposal of the traffic situation modelling, and in Subsect. 2.3 we talk about generation of explanations. The framework is illustrated by a detailed use case in Sect. 3. Finally, we conclude in Sect. 4.

2 Modelling Traffic Situations

2.1 Ontology

We use an ontology, illustrated in Fig. 1, to design the situation model. The ontology is based on Matheus, Kokar and Baclawski [12]. This work provides a more thorough explanation. The central part of this ontology is the *SituationObject* which can be either a *PhysicalObject*, e.g. *Car*, or a *NonPhysicalObject*, e.g. *Lane*. *Situation* extends *SituationObject* which allows to model a situation as an aggregation of sub-situations. *SituationObject* can have multiple instances of *Attribute* like speed or brake pressure of a car. In contrast to *Attribute*, which is specific to a *SituationObject*, *Relation* defines a relation between two situation objects. For example *CarA* is located on (*isOn*) *LaneX*. However, the values of

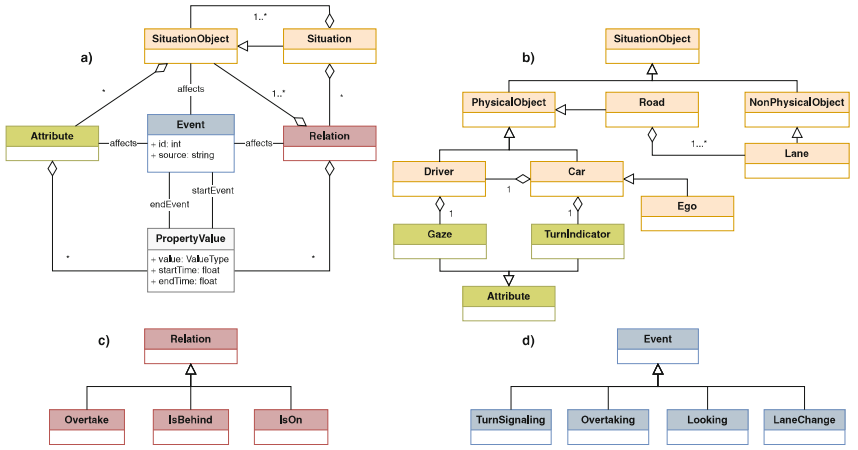


Fig. 1. (a) The core ontology introduced by Matheus *et al.* [12] is the basis of the domain-specific ontology proposed in this work. The expansion of (b) *SituationObject* and *Attribute*, (c) *Relation* and (d) *Event*. The concepts are expanded to sufficiently cover a simplified overtake scenario, see Sect. 3.

attributes and relations can change over time. This is captured by the *PropertyValue* concept.

Individual attributes and relations do not have a single value assigned to them, but multiple instances of *PropertyValue* with non-overlapping time intervals. The *Event* concept indicates a change of an attribute, relation, or more abstractly change in the situation, e.g. a change of *isOn* relation from *LaneX* to *LaneY* would generate a *LaneChange* event. A new *PropertyValue* is created whenever a new *Event* occurs.

In cases where we want to reason back in time, the actual event happens at or after the end time of another property value. We extend the core ontology of [12] in three aspects to be able to reason back in time. 1) The start and end time of a *PropertyValue* is not automatically set by the time of the current and next event. They can be inferred and set individually, contrasting to the core ontology of [12] where two temporally adjacent instances of *Event* always set the *StartTime* and the *EndTime* of a *PropertyValue*. 2) Start and end time need to be non-overlapping, but there can be a gap in between the end time of one property value and the start time of the next property value. 3) Situation objects, attributes and relations can be added dynamically once they get relevant for the situation.

As traffic situations can get very complex, the provided version of ontology is simplified and consists only of the objects relevant for the presented overtake scenario, shown in Sect. 3.

2.2 Multi-agent System

We employ a multi-agent system to assess all situations which arise during a lesson. Each agent performs specified tasks and communicates with the others through a blackboard, which stores all the data generated by agents. We differentiate between two types of agents: property value and explanation. A property value agent produces property values in the form of attributes or relations, and generates events accordingly when these attributes and relations change. Additionally, a property value agent can dispatch execution triggers to other agents. An explanation agent assesses if the student behaves correctly in a situation, and generates situation-specific explanations. In Fig. 2 we show a schematic of our suggested multi-agent system. It is constrained to assess a simplified overtake scenario for clarity. The agents are arranged in a hierarchical structure. Low-level agents perform basic assessment, such as checking the gaze of the driver or which lane the car is on. High-level agents can perform comprehensive tasks, for example assessing a complete overtake manoeuvre using data generated by many lower-level agents. Most explanation agents are thus higher-level agents.

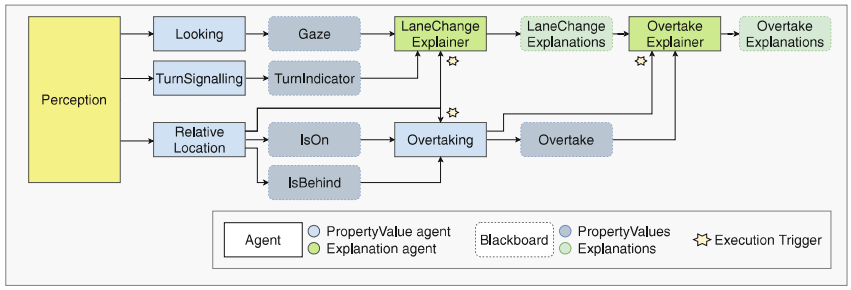


Fig. 2. The schematic of the proposed multi-agent system, simplified for a reduced overtake scenario. The *RelativeLocationAgent* dispatches execution triggers whenever a lane change has occurred.

2.3 Generating Explanations

Explanations play an important part in teaching. High quality explanations creates trust and motivate students. Our goal is to design a framework which generates explanations about both right and wrong behaviour at traffic situations encountered by students. This requires a system that has the domain knowledge to be able to assess the situation and the decisions the student made.

In our architecture, the domain knowledge is encoded within the ontology and the agents. Explanation agents are triggered by property value agents in order to infer if traffic rules were followed and if the driver behaved properly in the given situation. The output of an explanation agent is a recursive vector

structure containing all information necessary, also from more low-level explanation agents, for feedback generation. Each entry of the output vector contains an appropriate value or another vector which is the output of a subordinate agent. This structure allows to give explanations at any level of detail and is by itself already explainable. But the task of generating feedback for the student requires appropriate conversion of this data structure to a human-understandable form. The type of feedback which is suitable in each particular situation depends on whether it should be given online (while the lesson is running) or offline (as a debriefing session after the lesson). The online feedback is typically given as a short audio statement or an icon overlay, or via objects in the simulation itself. The offline feedback which is not time-sensitive can be more detailed. A detailed explanation text or an annotated video recording are examples of offline feedback. In this work, we focus on natural language feedback provided in an offline setting. The explanation vectors we propose have clearly defined structure and relatively small range of possible values. This allows us to use a simple and more robust template approach [6]. An example on how to generate text from explanation vectors is given in Subsect. 3.3.

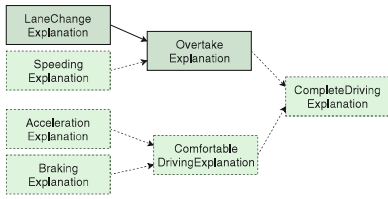


Fig. 3. Explanation tree structure illustrating that all explanations culminate into a high level *CompleteDrivingExplanation*. The figure shows the possibility of expanding the tree when new explanations are needed. It is not restricted to a binary tree. In this paper, we use only the *LaneChangeExplanation* and the *OvertakeExplanation*.

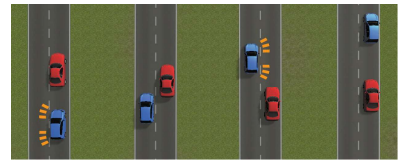


Fig. 4. Visualisation of the progress of the scenario, from left to right: indicate the intention to overtake ⟶ change lanes and pass the car ⟶ indicate the intention to return to the original lane ⟶ change back to the original lane.

3 Scenario: Overtake

3.1 Scenario Development

In the following, we outline a scenario in which the Ego car, driven by the driving student, overtakes another car, as depicted in Fig. 4. For this simplified case, we concentrate on the correct usage of turn signals and side mirror observation.

While the scenario develops over time, the situation is observed by adding property values to the ontology describing the complete situation. As stated

before, new property values record relations between situation objects and attributes of situation objects within a time period. Property values for relations with a boolean value and attributes are denoted as

$$< relation > (SituationObject_x, SituationObject_y, < start\ time >)$$
$$< attribute > (SituationObject_x, < start\ time >) :=< value >$$

Scenario:

- *Event* t_0: Start situation
  - $IsOn(\mathbf{Ego}, \mathbf{RightLane}, t_0)$
  - $IsOn(\mathbf{Car}, \mathbf{RightLane}, t_0)$
  - $IsBehind(\mathbf{Ego}, \mathbf{Car}, t_0)$
- *Event* t_1: Signalling left turn
  - $TurnIndicator(\mathbf{Ego}, t_1)$:=Left
- *Event* t_2: Looking at left mirror
  - $Gaze(\mathbf{Driver}, t_2)$:=LeftMirror
- *Event* t_3: Changing lanes
  - $IsOn(\mathbf{Ego}, \mathbf{LeftLane}, t_3)$
- *Event* t_4: In front of other Car
  - $IsBehind(\mathbf{Car}, \mathbf{Ego}, t_4)$
- *Event* t_5: Signalling right turn
  - $TurnIndicator(\mathbf{Ego}, t_5)$:=Right
- *Event* t_6: Looking at right mirror
  - $Gaze(\mathbf{Driver}, t_6)$:=RightMirror
- *Event* t_7: Changing lanes
  - $IsOn(\mathbf{Ego}, \mathbf{RightLane}, t_7)$
  - $Overtake(\mathbf{Ego}, \mathbf{Car}, t_7)$
- *Event* t_8: End situation
  - $TurnIndicator(\mathbf{Ego}, t_8)$:=Off

3.2 Reasoning Using First Order Logic

To identify state changes in attributes and relations, agents uses first-order logic as the predicates shown below.

$$\neg IsOn(\mathbf{Ego}, \mathbf{Lane}, t-1) \wedge IsOn(\mathbf{Ego}, \mathbf{Lane}, t) \implies LaneChange(\mathbf{Ego}, \mathbf{Lane}, t)$$

$$\neg\,(TurnIndicator(\mathbf{Ego}, t-1) = \text{Left}) \wedge (TurnIndicator(\mathbf{Ego}, t) = \text{Left}) \implies TurnSignalling(\mathbf{Ego}, \text{Left}, t)$$

The *OvertakeAgent* has one task – to recognise an overtake event. An overtake should be checked every time Ego does a lane change. The *LaneChange* event triggers the execution of the *OvertakeAgent*. Additionally, we query about the temporal data.

$$LOT(car, lane, t_{cur}) \equiv \underset{\forall t : \exists IsOn(car, lane, t) \wedge t < t_{cur}}{\operatorname{argmin}} (t_{cur} - t) \tag{1}$$

To know if an overtake occurred, one would like to know when the last occurrence in time (*lot*) where Ego was in the same lane as the one it is currently on. We

can find *lot* using Eq. 1, $lot = LOT(\mathbf{Ego}, \mathbf{Lane}, t)$, and input it into the overtake rule as follows.

$$IsOn(\mathbf{Ego}, \mathbf{Lane}, t) \wedge IsOn(\mathbf{Car}, \mathbf{Lane}, t) \wedge IsBehind(\mathbf{Car}, \mathbf{Ego}, t) \wedge$$
$$IsOn(\mathbf{Ego}, \mathbf{Lane}, lot) \wedge IsOn(\mathbf{Car}, \mathbf{Lane}, lot) \wedge IsBehind(\mathbf{Ego}, \mathbf{Car}, lot)$$
$$\implies Overtake(\mathbf{Ego}, \mathbf{Car}, t)$$

3.3 Explanations

By explicitly utilising explanation agents, one is able to separate the situation comprehension and the situation explanation. For instance, the *RelativeLocationAgent* recognises that a lane change has occurred, but does not care about how well the lane change was done. That task is delegated to the *LaneChangeExplainerAgent*.

The *RelativeLocationAgent* notifies the *LaneChangeExplainerAgent*, and the explanation agent assesses if the driver remembered to conduct all of the important actions needed for a proper lane change. To accomplish this, it need access to attributes from the past such as the property values *TurnIndicator* and *Gaze*. To confirm that the driver looked at the correct mirrors and switched on the turn signals prior to the time of the lane change, $t_{LaneChange}$, we define a time interval of interest as $I \equiv [t_{LaneChange} - i, t_{LaneChange}]$. Here i is a natural number defining the duration of the time interval in abstract units. Additionally, one could check if the sequence of these behaviours were correct, but in this example the explanation will be kept on a basic level. The *LaneChangeExplainerAgent* checks the following rules whenever a lane change occurs.

$$\text{SignalLeft} := \exists\, TurnIndicator(\mathbf{Ego}, t) = \text{Left} : t \in I$$
$$\text{LeftMirrorLook} := \exists\, Gaze(\mathbf{Driver}, t) = \text{LeftMirror} : t \in I$$
$$\text{LaneChangeExplanation} := \{\text{SignalLeft}, \text{LeftMirrorLook}\}$$

The results of these predicates are stored in the *LaneChangeExplanation* vector. Hence, given a lane change explanation, $LCE := \{\text{True}, \text{False}\}$, a template approach can be applied in the natural language generation.

> You performed a lane change.
> You did turn signal < *$TurnIndicator* ? correctly : incorrectly >,
> and you did < *$SideMirrorLook* ? check : not check >
> for cars behind you in the side mirror.

The explanation tree, illustrated in Fig. 3, shows that our multi-agent system can provide a complete explanation of a high level situation using the recursive vector structure. Multiple text snippets, from each explanation, can be merged to form a detailed human readable explanation. In this case, the *OvertakeExplainerAgent* has access to the *LaneChangeExplanations*. By retrieving the two

lane change explanations (LCE_1 and LCE_2) which defined the overtake, one can explain the complete overtake process.

$$\text{OvertakeExplanation} \equiv \{LCE_1, LCE_2\}$$

4 Conclusion

We have developed an architecture for a virtual driving instructor system, which can assess complex situations, such as the presented overtake scenario, and can derive conclusions or explanations of interest.

The multi-agent system allows the VDI to recognise traffic regulation violations as well as correct traffic behaviour. The explanation data structure generated by the multi-agent system has all the information necessary to generate complete, interpretable and traceable explanations. An example of such explanation using templates is also shown for the example of an overtake scenario.

As this work focuses on architectural design of an ITS, its implementation is a necessary next step in this project. An integrated ITS also requires a detailed design of the student model and its relation with the personalized feedback concept. Development of the student model is also left as a future work.

References

1. Arroyo, E., Sullivan, S., Selker, T.: CarCOACH: a polite and effective driving COACH. In: Proceedings of the Conference on Human Factors in Computing Systems, pp. 357–362 (2006). https://doi.org/10.1145/1125451.1125529
2. Backlund, P., Engström, H., Johannesson, M., Lebram, M.: Games for traffic education: an experimental study of a game-based driving simulator. Simul. Gaming **41**(2), 145–169 (2010). https://doi.org/10.1177/1046878107311455
3. Buechel, M., Hinz, G., Ruehl, F., Schroth, H., Gyoeri, C., Knoll, A.: Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles. In: 2017 IEEE Intelligent Vehicles Symposium (IV), vol. 7, pp. 1471–1476. IEEE, June 2017. https://doi.org/10.1109/IVS.2017.7995917
4. Chen, B., Cheng, H.H.: A review of the applications of agent technology in traffic and transportation systems. IEEE Trans. Intell. Transp. Syst. **11**(2), 485–497 (2010). https://doi.org/10.1109/TITS.2010.2048313
5. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Hum. Factors **37**(1), 32–64 (1995). https://doi.org/10.1518/001872095779049543
6. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks, applications and evaluation. CoRR abs/1703.09902 (2017)
7. Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: an overview of interpretability of machine learning. In: Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics. DSAA 2018, pp. 80–89 (2019). https://doi.org/10.1109/DSAA.2018.00018
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)

9. Gunning, D., Aha, D.W.: DARPA's explainable artificial intelligence program. AI Mag. **40**(2), 44–58 (2019)
10. Gutierrez, G., Iglesias, J.A., Ordoñez, F.J., Ledezma, A., Sanchis, A.: Agent-based framework for advanced driver assistance systems in urban environments. In: FUSION 2014–17th International Conference on Information Fusion (2014)
11. Hagras, H.: Toward human-understandable, explainable AI. Computer **51**(9), 28–36 (2018). https://doi.org/10.1109/MC.2018.3620965
12. Matheus, C.J., Kokar, M.M., Baclawski, K.: A core ontology for situation awareness. In: Proceedings of the 6th International Conference on Information Fusion, FUSION 2003, vol. 1, pp. 545–552 (2003). https://doi.org/10.1109/ICIF.2003.177494
13. McAree, O., Aitken, J.M., Veres, S.M.: Towards artificial situation awareness by autonomous vehicles. IFAC-PapersOnLine **50**(1), 7038–7043 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1349
14. Meiring, G.A.M., Myburgh, H.C.: A review of intelligent driving style analysis systems and related artificial intelligence algorithms. Sensors (Switzerland) **15**(12), 30653–30682 (2015). https://doi.org/10.3390/s151229822
15. Oulhaci, M.A., Tranvouez, E., Espinasse, B., Fournier, S.: Intelligent tutoring systems and serious game for crisis management: a multi-agents integration architecture. In: Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, pp. 253–258 (2013). https://doi.org/10.1109/WETICE.2013.78
16. Raptis, D., Iversen, J., Mølbak, T.H., Skov, M.B.: Dara: assisting drivers to reflect on how they hold the steering wheel. In: ACM International Conference Proceeding Series, pp. 1–12 (2018). https://doi.org/10.1145/3240167.3240186
17. Sipele, O., Zamora, V., Ledezma, A., Sanchis, A.: Advanced driver's alarms system through multi-agent paradigm. In: 2018 3rd IEEE International Conference on Intelligent Transportation Engineering. ICITE 2018, pp. 269–275 (2018). https://doi.org/10.1109/ICITE.2018.8492600
18. Sukthankar, R., Hancock, J., Pomerleau, D., Thorpe, C.: A simulation and design system for tactical driving algorithms. In: Proceedings of AI, Simulation and Planning in High Autonomy Systems, vol. 6 (1996)
19. Weevers, I., Kuipers, J., Brugman, A.O., Zwiers, J., van Dijk, E.M.A.G., Nijholt, A.: The virtual driving instructor creating awareness in a multiagent system. In: Xiang, Y., Chaib-draa, B. (eds.) AI 2003. LNCS, vol. 2671, pp. 596–602. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44886-1_56
20. Nkambou, R., Bourdeau, J., Mizoguchi, R.: Advances in Intelligent Tutoring Systems. Studies in Computational Intelligence. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14363-2
21. Zamora, V., Sipele, O., Ledezma, A., Sanchis, A.: Intelligent agents for supporting driving tasks: an ontology-based alarms system. In: VEHITS 2017 - Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems, pp. 165–172 (2017). https://doi.org/10.5220/0006247601650172
22. Zhao, L., Ichise, R., Yoshikawa, T., Naito, T., Kakinami, T., Sasaki, Y.: Ontology-based decision making on uncontrolled intersections and narrow roads. In: Proceedings of the IEEE Intelligent Vehicles Symposium, vol. 2015-Augus, pp. 83–88. IEEE (2015). https://doi.org/10.1109/IVS.2015.7225667

Bibliography

[1] S. Gregor and A. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quarterly*, vol. 37, pp. 337–356, 06 2013.

[2] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, I. Thomas, and K. Yelick, "A case for intelligent ram: Iram," 03 1997.

[3] M. West, "Evolve your hierarchy," January 2007. [Online]. Available: http://cowboyprogramming.com/2007/01/05/evolve-your-heirachy/

[4] C. Matheus, M. Kokar, and K. Baclawski, "A core ontology for situation awareness," 02 2003, pp. 545– 552.

[5] P. E. Santos, G. Ligozat, and M. Safi-Samghabad, "An occlusion calculus based on an interval algebra," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, 2015, pp. 128–133.

[6] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[7] M. K. H. Sandberg, "Design and implementation of the architecture of a virtual driving instructor," Master's thesis, Norwegian University of Science and Technology, NTNU, 7 2020.

[8] V. Vaishnavi and B. Kuechler, "Design science research in information systems," *Association for Information Systems*, 01 2004.

[9] B. Oates, *Researching Information Systems and Computing*. SAGE Publications, 2006.

[10] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013.

[11] R. Fabian, *Data-oriented design: software engineering for limited resources and short schedules.* Richard Fabian, 2018.

[12] K. Scott, "On proebsting"s law," USA, Tech. Rep., 2001.

[13] R. Nystrom, *Game Programming Patterns.* Genever Benning, 2014. [Online]. Available: https://gameprogrammingpatterns.com/

[14] T. Fontana, R. Netto, V. Livramento, C. Guth, S. Almeida, L. Pilla, and J. L. Güntzel, "How game engines can inspire eda tools development: A use case for an open-source physical design library," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, ser. ISPD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 25–31. [Online]. Available: https://doi.org/10.1145/3036669.3038248

[15] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. Wiley Publishing, 2009.

[16] M. Hadzic, E. Chang, and P. Wongthongtham, *Ontology-Based Multi-Agent Systems.* Springer Publishing Company, Incorporated, 2014.

[17] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[18] P. Hoen, K. Tuyls, L. Panait, S. Luke, and J. Poutré, "An overview of cooperative and competitive multiagent learning." 01 2005, pp. 1–46.

[19] M. Endsley, "Endsley, m.r.: Toward a theory of situation awareness in dynamic systems. human factors journal 37(1), 32-64," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, pp. 32–64, 03 1995.

[20] D. Fensel, U. Simsek, K. Angele, E. Huaman, K. Elias, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler, *Knowledge Graphs - Methodology, Tools and Selected Use Cases*, D. Fensel, Ed. Springer, 2020.

[21] A. W. Brown, J. Conallen, and D. Tropeano, *Introduction: Models, Modeling, and Model-Driven Architecture (MDA).* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–16. [Online]. Available: https://doi.org/10.1007/3-540-28554-7_1

[22] A. Metzger, *A Systematic Look at Model Transformations.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–33. [Online]. Available: https://doi.org/10.1007/3-540-28554-7_2

[23] K. Geihs, P. Baer, R. Reichle, and J. Wollenhaupt, "Ontology-based automatic model transformations," 01 2008, pp. 387–391.

[24] L. Vila, "A survey on temporal reasoning in artificial intelligence," *AI Commun.*, vol. 7, no. 1, p. 4–28, Mar. 1994.

[25] A. K. Pani and G. P. Bhattacharjee, "Temporal representation and reasoning in artificial intelligence: A review," *Math. Comput. Model.*, vol. 34, no. 1–2, p. 55–80, Jul. 2001. [Online]. Available: https://doi.org/10.1016/S0895-7177(01)00049-8

[26] J. McCarthy and P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, 1969, pp. 463–502.

[27] J. F. Allen, "Towards a general theory of action and time," *Artificial Intelligence*, vol. 23, no. 2, pp. 123 – 154, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370284900080

[28] R. Kowalski and M. Sergot, *A Logic-Based Calculus of Events*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 23–55. [Online]. Available: https://doi.org/10.1007/978-3-642-83397-7_2

[29] M. Krötzsch, F. Simancik, and I. Horrocks, "A description logic primer," *CoRR*, vol. abs/1201.4089, 2012. [Online]. Available: http://arxiv.org/abs/1201.4089

[30] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook*, 2nd ed. Cambridge, UK: Cambridge University Press, 2007.

[31] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *CoRR*, vol. abs/1809.02627, 2018. [Online]. Available: http://arxiv.org/abs/1809.02627

[32] P. Jayanti, T. D. Chandra, and S. Toueg, "Fault-tolerant wait-free shared objects," *J. ACM*, vol. 45, no. 3, p. 451–500, May 1998. [Online]. Available: https://doi.org/10.1145/278298.278305

[33] M. Herlihy and N. Shavit, "On the nature of progress," in *Principles of Distributed Systems*, A. Fernàndez Anta, G. Lipari, and M. Roy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 313–328.

[34] S. Peri, C. K. Reddy, and M. Sa, "An efficient practical concurrent wait-free unbounded graph," in *2019 IEEE 21st International Conference on*

High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Aug 2019, pp. 2487–2494.

[35] N. D. Matsakis and F. S. Klock, "The rust language," *Ada Lett.*, vol. 34, no. 3, p. 103–104, Oct. 2014. [Online]. Available: https://doi.org/10.1145/2692956.2663188

[36] S. Klabnik and C. Nichols, *The Rust Programming Language*. USA: No Starch Press, 2018.

[37] A. Antelmi, G. Cordasco, M. D'Auria, D. De Vinco, A. Negro, and C. Spagnuolo, "On evaluating rust as a programming language for the future of massive agent-based simulations," in *Methods and Applications for Modeling and Simulation of Complex Systems*, G. Tan, A. Lehmann, Y. M. Teo, and W. Cai, Eds. Singapore: Springer Singapore, 2019, pp. 15–28.

[38] T. Dutoit, "High-quality text-to-speech synthesis : an overview," *Journal of Electrical & Electronics Engineering*, 1997.

[39] S. Mache, M. Baheti, C. Mahender, and A. Professor, "Review on text-to-speech synthesizer," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, pp. 54–59, 09 2015.

[40] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," 2017.

[41] A. Doniec, R. Mandiau, S. Piechowiak, and S. Espié, "A behavioral multi-agent model for road traffic simulation," *Eng. Appl. Artif. Intell.*, vol. 21, no. 8, p. 1443–1454, Dec. 2008. [Online]. Available: https://doi.org/10.1016/j.engappai.2008.04.002

[42] T. Bosse, M. Hoogendoorn, M. Klein, and J. Treur, "A component-based ambient agent model for assessment of driving behaviour," 06 2008, pp. 229–243.

[43] F. E. Garcia and V. P. de Almeida Neris, "A data-driven entity-component approach to develop universally accessible games," 2014, pp. 229–243.

[44] G. Gutierrez, J. A. Iglesias, F. J. Ordoñez, A. Ledezma, and A. Sanchis, "Agent-based framework for advanced driver assistance systems in urban environments," in *17th International Conference on Information Fusion (FUSION)*, July 2014, pp. 1–8.

[45] M. Danielsson and G. P. Bohlin, "A high performance data-driven, entity-component framework for game engines with focus on data-oriented design," 2015.

[46] T. Nguyen, "A multi-agent architecture for situation awareness," in *Proceedings of 1st International Conference on Conventional and Knowledge Based Intelligent Electronic Systems. KES '97*, vol. 2, May 1997, pp. 502–507 vol.2.

[47] P. Morignot and F. Nashashibi, "An ontology-based approach to relax traffic regulation for autonomous vehicle assistance," 12 2012.

[48] H. Toulni, B. Nsiri, M. Boulmalf, and T. Sadiki, "An ontology based approach to traffic management in urban areas," 2015.

[49] L. Zhao, R. Ichise, S. Mita, and Y. Sasaki, "An ontology-based intelligent speed adaptation system for autonomous cars," in *Semantic Technology*, T. Supnithi, T. Yamaguchi, J. Z. Pan, V. Wuwongse, and M. Buranarach, Eds. Cham: Springer International Publishing, 2015, pp. 397–413.

[50] D. D. Hodson and J. R. Millar, "Application of ecs game patterns in military simulators," *Proceedings of the International Conference on Scientific Computing (CSC)*, 2018.

[51] J. A. Sharp, "Data oriented program design," *SIGPLAN Not.*, vol. 15, no. 9, p. 44–57, Sep. 1980. [Online]. Available: https://doi.org/10.1145/947706.947713

[52] R. Weiss and C. Steger, "Design and implementation of a real-time multi-agent system," in *MELECON '98. 9th Mediterranean Electrotechnical Conference. Proceedings (Cat. No.98CH36056)*, vol. 2, May 1998, pp. 1269–1273 vol.2.

[53] K. Barber, N. Gujral, J. Ahn, D. DeAngelis, K. Fullam, D. Han, D. Lam, and J. Park, "Design, runtime, and analysis of multi-agent systems," 01 2005, pp. 157–158.

[54] D. Hall, "Ecs game engine design," 06 2014.

[55] X. Su, H. Cai, B. Luong, and S. Ukkusuri, "From a link-node-based network representation model to a lane-based network representation model: Two-dimensional arrangements approach," *Journal of Computing in Civil Engineering*, vol. 29, p. 04014045, 10 2014.

[56] P. Lange, R. Weller, and G. Zachmann, "Graphpool: A high performance data management for 3d simulations," 05 2016, pp. 23–33.

[57] S. Fuchs, S. Rass, and K. Kyamakya, "Integration of ontological scene representation and logic-based reasoning for context-aware driver assistance systems," *ECEASST*, vol. 11, 01 2008.

[58] M. A. Oulhaci, E. Tranvouez, B. Espinasse, and S. Fournier, "Intelligent tutoring systems and serious game for crisis management: A multi-agents integration architecture," in *2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 2013, pp. 253–258.

[59] R. Sukthankar, S. Baluja, and J. Hancock, "Multiple adaptive agents for tactical driving," *Applied Intelligence*, vol. 9, 07 2002.

[60] A. Armand, D. Filliat, and J. Ibanez-Guzman, "Ontology-based context awareness for driving assistance systems," 06 2014, pp. 227–233.

[61] L. Zhao, R. Ichise, T. Yoshikawa, T. Naito, T. Kakinami, and Y. Sasaki, "Ontology-based decision making on uncontrolled intersections and narrow roads," 06 2015.

[62] M. A. Mohammad, I. Kaloskampis, Y. Hicks, and R. Setchi, "Ontology-based framework for risk assessment in road scenes using videos," *Procedia Computer Science*, vol. 60, pp. 1532 – 1541, 2015, knowledge-Based and Intelligent Information and Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915024278

[63] F. Fang, S. Yamaguchi, and A. Khiat, "Ontology-based reasoning approach for long-term behavior prediction of road users," 10 2019, pp. 2068–2073.

[64] A. Bermejo, J. Villadangos, J. Astrain, and A. Cordoba, "Ontology based road traffic management," vol. 1-2, 09 2012.

[65] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, and A. Knoll, "Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1471–1476.

[66] D. Krol and F. Nowakowski, "Practical performance aspects of using real-time multi-agent platform in complex systems," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2013, pp. 1121–1126.

[67] Jinhuan Wang and Baomin Li, "Study of semantic reasoning based on ontology description logic," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Oct 2016, pp. 1869–1872.

[68] Y. Shoham, "Temporal logics in ai: Semantical and ontological considerations," *Artificial Intelligence*, vol. 33, no. 1, pp. 89 – 104, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/000437028790052X

[69] U. Demiryurek, B. Pan, F. Banaei-Kashani, and C. Shahabi, "Towards modeling the traffic data on road networks," in *Proceedings of the Second International Workshop on Computational Transportation Science*, ser. IWCTS '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 13–18. [Online]. Available: https://doi.org/10.1145/1645373.1645376

[70] M. Hulsen, J. M. Zöllner, and C. Weiss, "Traffic intersection situation description ontology for advanced driver assistance," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 993–999.

[71] D. Schmalstieg, "Unified patterns for realtime interactive simulation in games and digital storytelling," *IEEE Computer Graphics and Applications*, vol. 39, no. 1, pp. 100–106, Jan 2019.

[72] R. Regele, "Using ontology-based traffic models for more efficient decision making of autonomous vehicles," in *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, March 2008, pp. 94–99.

[73] N. D. Kallimanis and E. Kanellou, "Wait-Free Concurrent Graph Objects with Dynamic Traversals," in *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), E. Anceaume, C. Cachin, and M. Potop-Butucaru, Eds., vol. 46. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 1–17. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6616

[74] P. Lange, R. Weller, and G. Zachmann, "Wait-free hash maps in the entity-component-system pattern for realtime interactive systems," in *2016 IEEE 9th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, March 2016, pp. 1–8.

[75] M. Musen, "The protégé project," *AI Matters*, vol. 1, pp. 4–12, 06 2015.

[76] J. Gjengset, M. Schwarzkopf, J. Behrens, L. T. Araújo, M. Ek, E. Kohler, M. F. Kaashoek, and R. Morris, "Noria: dynamic, partially-stateful data-flow for high-performance web applications," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 213–231. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/gjengset

[77] J. Wang, "Scheduling of periodic tasks with data dependency on multiprocessors," *Advanced Materials Research*, vol. 756-759, 05 2014.

[78] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[79] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert scale: Explored and explained," *British Journal of Applied Science & Technology*, vol. 7, pp. 396–403, 01 2015.

Appendices

Appendix A: Interview questions

#	Question
#1	How was the experience in general? Was it positive or negative? Did you enjoy it?
#2	How does the system perform in comparison with your expectations? Was it worse or better? Why?
#3	What do you think about the feedback content? Was it easy to understand what mistake you made? Was it clear how to avoid the mistake the next time? Was there any information missing? Would you change something?
#4	How do you feel about the feedback timing? Was it appropriate? Was it accurate? Why? Why not?
#5	Did you mind the feedback was audio only? Would you like to see other kinds/forms of feedback? Do you think they will be beneficial? What kind of forms?
#6	What was the assessment quality of each phenomenon? How good was: Incorrect gear, Speeding, Overtake, Yielding at traffic light controlled intersections, Yielding for pedestrians assessment? What was good about them? What was bad about them? Any other remarks, findings, opinions?
#7	How close do you think the system performs in comparison with human teachers? What was nearly or completely identical? What was not really human-like? Why?
#8	What do you think needs/can be done to improve the system? What on the other hand should stay the same? Why?
#9	Do you believe that a system like this should be worked on and developed further? Why yes, why not? Do you see any obvious deal-breakers already?
#10	Anything else we have not covered and you would like to address/mention?

Table 6.1: Questions used for semi-structured interviews carried out during Evaluation phase (see chapters 4 and 5).

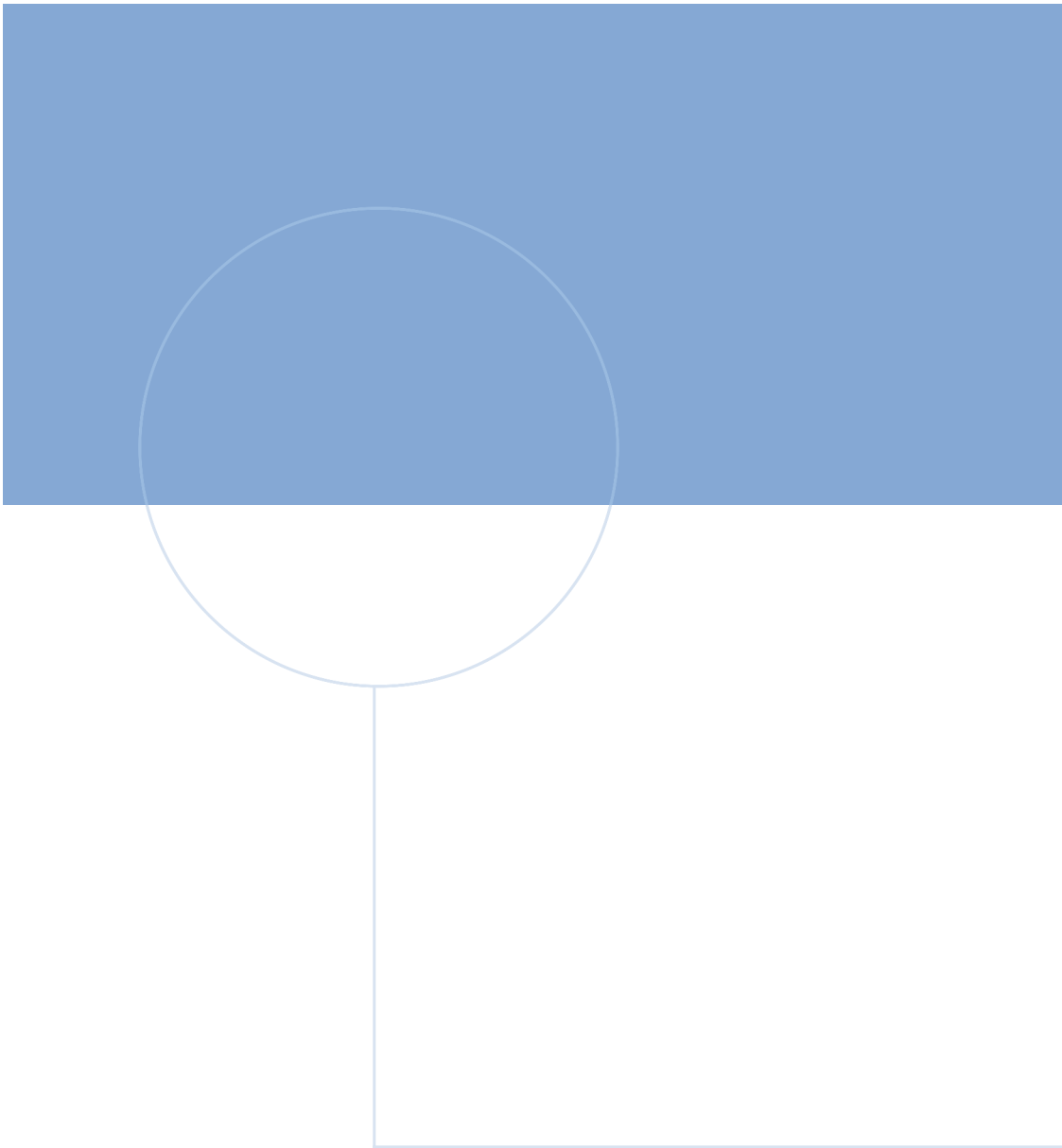
Appendix B: Questionnaire

#	Question	**Likert Scale** (1-7)
#1	How would you rate your general experience with the assessment system?	Very dissatisfied <-> Very satisfied
#2	How informative was the feedback overall?	Very non-informative <-> Very informative
#3	How good was the timing of the feedback?	Completely off <-> Very good
#4	How good was the feedback in terms of accuracy?	Completely off <-> Very good
#5	Was it easy to understand the voice?	Very hard and unclear <-> Very easy and clear
#6	How "human" was the feedback?	Very "non-human" and quite far from a real teacher behaviour <-> Very "human" and close to a real driving teacher
#7	How good was the assessment of correct/incorrect gear?	Very bad <-> Very good
#8	How good was the assessment of speeding?	Very bad <-> Very good
#9	How good was the assessment of overtakes?	Very bad <-> Very good
#10	How good was the assessment of yielding at traffic light controlled intersections?	Very bad <-> Very good
#11	How good was the assessment of yielding to pedestrians at crosswalks?	Very bad <-> Very good
#12	Do you believe that a system like this has potential and can be a helpful tool for simulator-based driving education?	Strongly disagree <-> Strongly agree
#13	Do you think that a system like this should be worked on and developed further?	Strongly disagree <-> Strongly agree
#14	Were there any issues with the system during your evaluation?	A lot <-> None

Table 6.2: Questionnaire used during Evaluation phase (see chapters 4 and 5).

Appendix C: Assessment system logs

The assessment system presented in this thesis is capable of logging its activity. Therefore, each of the evaluation runs with domain experts (see chapter 4) generated logs capturing the system operations. The logs were anonymized which makes it possible to publicly share them. The logs are available at the following link: Assessment system logs. For each lesson session performed during evaluation, one log file was generated. The log files are correspondingly named either *overtake* if they come from *Overtake lesson* and *intersection* if from *Traffic light controlled intersections lesson*. Moreover, each file is indexed so it is easy to pair logs which belong to the same domain expert.



NTNU

Norwegian University of
Science and Technology