./img/cvut-logo.jpg

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORT SCIENCES

JAN MACEK

AGENT-BASED MODELING OF ITS SYSTEMS IN VEHICLE SIMULATOR

DIPLOMA THESIS

2022

zadani.pdf

zadani.pdf

Abstract

TODO

Contents

1 Introduction

In a world where vehicle transport plays an inseparable role in the society, with an ever increasing demand, it is important to analyze and study driver behaviour and inherently interactions and relationships between drivers and their surroundings. This is supported by the fact that, even though substantial advancements in autonomous driving are being made, for the near-future, humans will still have to control vehicles themselves and therefore be exposed to a substantial risk of danger. Data shows that about 95 % of traffic accidents are a result of human error [1]. Each traffic accident has got a tremendous effect on socio-economic growth. A study be the European Union states that accident-related expenses (including cost of fatality) cost 1,8 % of EU GDP [2]. A rather non-cynical point of view is that each life lost is a failure in society itself and an effort should be made to diminish fatal accidents.

A method that has proven to be effective at studying driver behaviour and traffic safety is by using interactive vehicle simulators (IVS), which allow to undertake experiments in a safe, controlled and reproducible way. Because driving simulator is basically a digital twin of a real vehicle, it is naturally reasonable to make the interaction between the driver and IVS as close to reality as possible, which inherently improves data quality of the simulation and potentially also the range of IVS application. An IVS has got a broad spectrum of employment. It is not only used as a tool to research driver behaviour, but also used in development and testing of advanced driver-assistance systems, extending the simulator to a hardware-in-the-loop or a vehicle-in-the-loop system, which enables to test real hardware and simulate full road testing.

Simulating the traffic environment is a complex problem, mainly because of its highly dynamic characteristics. All vehicles need to interact with each other and act upon other drivers' actions. Because agent-based simulations have proven to model complex behaviour well, this modelling technique seems like a suitable solution for achieving a realistic traffic environment for IVS.

The goal of this thesis is to investigate multi-agent systems, their principles and evaluate usages of these systems related to ITS research, where their shared characteristics of distributed interoperability could prove to make agent-based modelling strong tool for simulating ITS.

After investigation of ABM in the ITS field, an experimental work is presented, which presents a generic framework that can be used to implement agent-based ITS into an IVS. This framework has got a self-proposed architecture in line with MAS paradigms, making it easier to streamline building a distributed ITS simulation of choice. Afterwards, a validation of the proposed framework is conducted by implementing a distributed ITS using said framework into an IVS, evaluating its performance and usability for future research.

2 Intelligent Transport Systems

Intelligent transport systems describe an initiative to utilize modern technology in order to optimize and increase efficiency of processes common to the domain of transportation. This usually involves enabling different actors in the transportation system to communicate with each other and share available information (much like multi-agent systems). With the availability of shared information, complex, data-driven systems can be built utilizing state-of-the-art tech concepts like machine learning, computer vision and IoT, while integrating humans, roads and automobiles, improved street protection, efficiency and stability. İt also address environmental issues via controlling traffic to prevent congestion or lessen their adverse effects.

When looked at these solution from a systemic point of view, they can be, in less or more ways, resemble multi-agent systems. This section will be focused more on the *inductive* part of research, examining various existing ITS solutions and trying to recognize their agent-based characteristics and principles, if there are any. Furthermore, specific systems will be identified, which could be used as a validation for the *deductive* approach using theory discussed later (section 3), where literature review of multi-agent systems will be undertaken. In other words, building a MAS-based simulation framework and then evaluating it using a chosen ITS. This will be discussed more in the following practical part (section 4).

`./img/ITS-schema.jpg`

Figure 1: Illustration of an ITS topology [3]

ITS can provide itself most useful in the following aspects of transportation: *Mobility*, *Safety*, *Environment*. To provide an example, regarding mobility, one of the most common systems is routing and navigation for road vehicles. Such systems can factor in time, distance and even emissions when providing route guidance. This leads to increased performance of road networks. Regarding safety, one can mention emergency management systems like E-Call, which provides automated post-crash assistance by contacting emergency services as soon as an incident happens. In terms of the environment, ITS allows for demand management through *electronic fee collection*, which makes flexible charging for road usage possible, based on vehicle type and emissions category [4]. Another, more general example of emissions reduction of ITS is through reducing congestions, as road vehicle emissions have proven to be a significant environmental factor not only in emission production per se, but also one that humans are most exposed to. In a study conducted by the Harvard School of Public Health, air pollution from traffic congestion in 83 of the USA's largest urban areas contributes to more than 2,200 premature deaths annually, costing the health care system at least $18 billion [5].

2.1 Implementation

In relation to the topic of this thesis, it is important to define ITS features that will create a qualitative analysis/benchmark for identifying suitable system(s) for the ITS implementation. The following sections will discuss selected features that will be used for deciding which system to implement as a demonstration of the proposed agent-based framework.

2.1.1 Driver engagement

Firstly, it is important to say that ITSs vary in terms of *driver engagement*. Because IVS's focus is to mainly research driver behaviour, a trivial requirement is to integrate with automotive transport. Furthermore, a system with a large degree of driver interaction should be chosen to have an observable effect.

As per (2), the ITS to develop/implement as part of the thesis having one or more of the following features should ensure high driver (i.e. user) engagement:

- Geo-info service
- Real-time road condition service
- Accurate traffic-info service
- Real-time vehicle info service
- Parking guidance service

./img/why is ITS needed.jpg

Figure 2: Reasons for ITS usage

2.1.2 MAS-compatibility

Secondly, it is important to acknowledge whether there is an ITS that could be simulated using the MAS theory, so that the information gathered in the first section of this thesis could be successfully applied.

Intelligent transport systems are, by nature, systems combining several actors together to achieve a common goal. Would this mean that every ITS can be, in fact, modelled as multi-agent system? The answer is that it does *not* apply to all of them.

Historically speaking, ITSs that were developed and proved helpful for optimizing traffic were *centralized*, meaning there is a single core in charge of logic and decision-making, organizing other units/actors within the system, while interacting with the drivers in the traffic as external actors. The main advantage of these systems is that they are less complex, therefore it is easier to develop a resilient and safe product. The main drawback of this method is that they are heavily affected by the size of instances, increasing the complexity and often result in exceedingly large computation time, making the solution sub-optimal [6].

As with the *distributed* systems, the main challenge making individual agents solving sub-problem, cooperating well to achieve good and predictable results, as has been illustrated in section (3). Distributed systems can be used to solve traffic optimization problems on a larger scale. These systems have seen larger usage especially in the recent times, as newly produced vehicles are equipped with powerful computers that can be used to transfer the computational burden from the core computer in centralized systems. Also state-of-art communication protocols like DSRC and ITS 5G, which enable low-latency direct communication between vehicles.

The one example of an ITS where implementations have been carried out in both centralized and decentralized principles numerous times is a *Network traffic control.* The goal of this system is to use knowledge about the traffic on the network to control traffic lights and other active traffic control elements to optimize traffic flow and decrease travel time. In [7], centralized and distributed network traffic control systems were compared. The conclusion was that although the centralized system was able to achieve better performance higher global efficiency, the decentralized solution required significantly less computation time (40 % in their experiment case).

Other decentralized, MAS-based systems that are more exposed to the user (driver) include the *Adaptive Cruise Control* (ACC), for example. ACC extends the usual cruise control systems that maintain vehicle speed by adapting to speed of the vehicle in front, if there is any. In fact, recent development has lead to a further improvement, making ACC a cooperative system (CACC) that enables vehicles to adopt a *driving strategy* by communicating with the infrastructure (V2I communication).

2.1.3 Current research

Secondly, it is important to choose a system that is time-relevant and would bring benefit to current as well as future IVS and HMI research activities. Therefore, the third quality for choosing an ITS to implement would be one that is still subject to research and current trends in the ITS industry and explore state-of-the-art ITS. In order to determine which ITSs are relevant, it is important to review ongoing project of governmental bodies and their strategies and also current trends in automotive ITS research activities.

In Europe, there are initiatives to centralize decision making and ITS deployment on the scale of the European continent. The reason behind this initiative is quite clear - to enable Europe-wide interoperability between deployed ITS and ITS-enabled vehicles. It is without a question that such organization would closely work with the industry, helping to create conditions for novel ITS technologies to be deployed.

ERTICO One such European organization is the *European Road Transport Telematics Implementation Coordination (ERTICO)* - a public-private partnership organization with close to 120 members, connecting 8 different sectors in the ITS Community, including service providers, suppliers, traffic and transport industry, research institutions and universities, public authorities, user organizations, connectivity industry as well as vehicle manufacturers [8].

As of now, ERTICO's activities focus on the following areas:

Connected Cooperative & Automated Mobility - As the computational power of newly-produced vehicles is increasing dramatically with every new generation, as well as the number of sensor data, ERTICO states their focus is on utilizing the large amounts of real-life data to deepen the machine learning models, as well as building an infrastructure that will allow handling this data. C-ITS (Cooperative Intelligent Transport Systems) is also mentioned, whose principles are being put into practice by several projects, namely European Truck Platooning (ETPC), Advanced map-enhanced driver assistance systems (ADASIS) and more. ERTICO's main contribution is to facilitate creation of ITS ecosystem by following a multidisciplinary approach involving all relevant transport stakeholder sectors.

Clean & Eco- Mobility - As has been already mentioned, smart mobility innovations make a major contribution towards reducing the impact of transport-induced pollution, which has got a non-negligible contribution to global greenhouse gas emissions production [9]. Below are the four main objectives in the are of Clean & Eco-Mobility.

- Develop a common approach to the evaluation of ITS deployment as a tool for emissions reduction
- Contribute to smart mobility solutions being recognized as a tool for reducing emissions
- Achieve interoperability of electro-mobility
- Contribute to creating an ICT network with seamless and interoperable electro-mobility services

Urban Mobility - Another focus of ERTICO is Urban Mobility, where the main goal is to provide "Mobility as a Service" (MaaS), which is a system that could decrease congestions and provide low-carbon and -emission multi-modal transport solutions.

Transport & Logistics - ERTICO states that the current European world of transport and logistics is too fragmented, so an effort to develop solution for connecting logistics information system would optimize cargo flows and facilitate supply chain management.

In conclusion, the ERTICO organization helps to reduce time to market of innovative, state-of-the-art technologies, increasing inter-operability between individual ITS by promoting an open framework for integration and deployment of intelligent transport services.

As described above, there are potential systems that are yet to be fully implemented and deployed for consumer use.

Cooperative ITS Cooperative Intelligent Transport Systems (C-ITS) refers to transport systems, where ITS subsystems (personal, vehicle, roadside and central) cooperate. This enables and provides an ITS service that offers better quality and an enhanced service level, compared to the same ITS service provided by only one of the ITS sub-systems [10]. An example of the concept of the technology can be seen in the figure (3) below.

./img/c-its.jpg

Figure 3: Example schematic of C-ITS scenario

The concept of C-ITS was developed by the European Commission, representatives of industry and authorities in the European Union. In 2016, it was agreed on a coordinated establishment of intelligent transport systems in Europe. It is considered to be one of the various tools to facilitate achieving the *vision zero*, which is a project that aims to mitigate all fatalities involving road transport.

The European Commission outlined its plan for the coordinated deployment of C-ITS in Europe in its communication 'A European strategy on Cooperative Intelligent Transport Systems', in which it also states that the full-scale deployment of C-ITS services and C-ITS enabled vehicles is expected to start in 2019.

The main feature of C-ITS is its distributed intelligence across vehicles and the infrastructure, which is a novel concept in the ITS world. Consequently, it is easy to see similarities to how MAS are described. Regarding the topic of this thesis, this could pose as an argument to implement one of C-ITS systems in the practical part of this thesis.

Vehicles and infrastructure equipped with C-ITS can, for example, communicate a warning to each other, after which the drivers are informed about the upcoming traffic situation in time for them to take the necessary actions in order to avoid potential harm. Other potential benefits of the use of C-ITS include reduced congestion and improved driver comfort. In short, vehicles share data directly between each other (V2V) and with the infrastructure (V2I) using ad-hoc short range telecommunication. The two types of communications are sometimes together referred to as Vehicle-to-everything (V2X) communication.

This technology aims to benefit both manually-driven vehicles as well as autonomous self-driving vehicles. The main use-cases, which were developed as standalone services using C-ITS technology can be seen in the figure (4) below.

./img/c-its-kolecka.png

Figure 4: C-ITS use cases [10]

In general, the traffic safety and traffic flow improvements can be grouped based on which operational tasks they serve [11]:

- Provide *information* to road users to improve road safety and comfort.
- Display *regulatory boundaries* to inform road users of specific obligations, restrictions or prohiitions
- Provide *warnings* to road users about incidents ahead in their exact nature.

Project C-Roads In order to study the effects and refine the future deployment process of C-ITS, a project organized by the EU member states and road operators, called *C-Roads* was established. The project's main objective is to harmonize C-ITS deployment activities across Europe. Within the C-Roads project, there have been established 5 work groups (WG), each focusing on a specific topic/scope regarding C-ITS objectives and priorities for research, testing and pre- deployment of C-ITS [12].

- WG1: Develop an EU agenda for testing
- WG2: Coordination and cooperation of R&I activities
- WG3: Physical and digital road infrastructure

- WG4: Road Safety
- WG5: Access and exchange of data & cyber-security
- WG6: Connectivity and digital infrastructure

Regarding the scope of this thesis, *WG3* is the most informative segment. The focus of this group was, in the first place, regarding the infrastructure support for automated vehicles (SAE L4). One of the objectives was linking relevant physical and digital infrastructure, assessing relevance between them and automated vehicles, and mapping infrastructure elements to use-cases as *generic driving tasks* (GDT):

- Sensing & Perception
  - Ego localization
  - Environmental awareness (object classification and incident detection)
  - Enhanced perception (for limited visibility scenarios)
- Planning
  - (dynamic) information and regulations
  - Safe and appropriate navigation plans
  - Cooperative planning
- Actuation
  - Motion Control
  - Minimum Risk Manoeuvre

It is therefore advisable to keep these GDT in mind when choosing which ITS to develop regarding the thesis, because previous R&I activities have shown that the tasks above will be a fundamental part of the future C-ITS deployment. Because all of the GDT seem to be more or less distinct activities, the table below (1) was created to map each GDT to particular C-ITS system, which will be used to narrow down the potential ITS implementation candidates and hopefully make the final decision for ITS implementation clear.

Table 1: GDT to ITS mapping

GDT		Mapping	
Group	Name	Type	Description
Sensing & Perception	Ego-localization	HD Maps	Geo-fencing
		AD	Self-driving algorithms
	Environmental awareness	IVIS	Intersection Collision war.
			Emergency Vehicle war.
			Stationary Vehicle war.
			Traffic Jam war.
			Traffic accident war.
	Enhanced perception	IVIS	Overtaking war.
			Intersection Collision war.
			VRU warning
Planning	Information and regulations	CACC	Green Light Optimal Speed Advisory
	Safe navigation plans	Navigation	Dynamic Vehicle Routing
	Cooperative planning	AD	Platooning
Actuation	Motion Control	AD	Self-driving algorithms
	Minimum Risk Manoeuvre	AD	Self-driving algorithms

Legend

Abbreviation	Meaning
HD	High-definition
AD	Autonomous driving
IVIS	In-Vehicle Information Systems
VRU	Vurneable Road User
CACC	Cooperative Adaptive Cruise Control

2.2 Qualitative analysis

Looking at the table (1), it is evident that all the tasks fall under *four* mapping types. Those will be used as entries in the qualitative analysis. Nevertheless, all types of ITS considered in the preceding sections will be examined and subsequently evaluated to decide which type of ITS would be the most optimal to implement.

The method to determine how well-suited the candidate system type is for the implementation to an IVS will be to evaluate it based on the *three* features discussed in the preceding sections:

- MAS-compatibility
- Driver engagement
- Research relevance

A set of integer values will determine significance of each feature. The significance will be given in four levels:

• None	→	0
• Low	→	2
• Moderate	→	5
• High	→	8

Table 2: Quantitative analysis results

	Features			
ITS	MAS-compatibility	Driver engagement	Research relevance	Score
E-Call	2	2	5	9
Electronic fee collection	2	0	2	4
Parking guidance	2	8	5	15
Network traffic control	8	0	8	16
Cooperative ACC	8	8	8	**24**
European Truck Platooning	5	2	8	15
ADASIS	2	8	8	18
Mobility as a Service	8	2	8	18
Map services (Geo-fencing)	2	5	8	15
Self-driving & Platooning algorithms	8	2	8	18
IVIS	8	8	8	**24**

2.3 In-Vehicle Information System

The In-Vehicle Information System, shortly IVIS, is a term that comprises a vast number of vehicle technology solutions that assist the driver either by providing information about the vehicle and the surrounding environment or serving as an interface to control vehicle systems.

The integral part is the *Infotainment system* (fig. 5), which is a video/audio interface, providing control through elements such as touch screen button panel or voice commands. Moreover, it integrates other vehicle technology elements, such as the CAN interface, connectivity modules (e.g. Wi-Fi, GPS), sensors etc [13]. A screen panel is an excellent medium to provide additional safety information to the driver, especially when the gauge clusters have been replaced by an additional screen which improves the HMI aspect by reducing the disruptive effect of checking the screen and making the displayed information more noticeable.

The conventional, more basic capabilities of IVIS are HVAC control, multimedia controls, navigation support and parking assistance (i.e. parking camera view). These systems on their own, however, aren't valuable with regard to the thesis topic. The important feature of IVIS is the integration with the emerging C-ITS technologies, which greatly extend the capabilities of IVIS, i.e. displaying warning and awareness messages from the V2X interface. This fact makes IVIS a good candidate to implement to the IVS in the practical part, because the V2X C-ITS are a great subject for research as of now and the distributed nature of the systems corresponds to the agent-based requirement of the thesis topic. On top of that, the important HMI aspect of IVIS could provide great value for future utilization in research using IVS.

The general idea for method of implementation is to implement C-ITS awareness and warning information system by simulating message-based communication between road users & infrastructure and provide interface to display the information on the infotainment screen.

2.4 Cooperative Adaptive Cruise Control

The Cooperative Adaptive Cruise Control (CACC) is an extension to an already well proven ITS - Adaptive Cruise Control. As has been described above, this system extends the base (adaptive) cruise control system by utilizing the V2V information broadcasted by other road users. The system has proved to reduce the number of shock-waves by reducing oscillations that would otherwise happen without speed information sharing and increasing capacity of the traffic network, albeit only with higher penetration rates ($\geq 40\%$) [14]. Though, regarding the introduced thesis topic requirements, the system in itself doesn't provide any extra engagement from the driver side.

./img/ivis-dashboard.jpg

Figure 5: Illustrative example of IVIS interface [13]

However, another system that built upon the concept of CACC by utilizing information from the infrastructure, namely the traffic signals, has been also recently developed this system can be extended by information fs about signal phasing and consequently adopting a speed that would eliminate the need to stop before a red light, avoiding idle time. The system is called *Green Light Optimization Speed Advisory* (GLOSA). According to [15], the results of simulating deployment of this system demonstrated that even using a simple control algorithm with the aim to avoid the stop&go a reduction of fuel consumption and emissions in the region of 5 to 12 % has been observed. The study in [16] even concluded that with enough penetration, the idle (stop) time could be decreased by more than 70% (see fig. 6 below).

The the proposed idea for simulating the system in an IVS would be to implement a GLOSA traffic controller algorithm and use the C-ITS information system to provide the driver with the optimal speed information, which would be displayed on the infotainment screen.

2.5 Conclusion

In this section, the Intelligent Transport Systems have been introduced. This field is an important part of traffic engineering, utilizing traffic data and mathematical modelling to reduce congestion, improve traffic safety and reduce emissions. The main point of this section was to find a suitable ITS to implement as a means for the MAS architecture validation proposed in this thesis. Three indicators for quantitative analysis have been determined (Driver engagement, MAS-compatibility, research relevance) and various ITS projects have been investigated, including the R&I efforts on the EU level. The results

./img/glosa-perf-chart.png

Figure 6: Idle time reduction based on GLOSA penetration rate [16]

suggested that C-ITS systems are a current trend in research and numerous projects (e.g. C-Roads) have been established, paving the way for the future of interconnected vehicle mobility. Afterwards, a methodology for qualitative analysis was introduced and used to determine the best ITS candidates for implementation into an IVS. The resulting best fitted systems were both from the C-ITS field, as its features are much alike those of multi-agent systems. Finally, the chosen systems to implement were the *IVIS based awareness and warning information system* and the *Green Light Optimal Speed Advisory* system. The following practical part will be dedicated to the implementation methodology, introduction to the development system and the development itself.

3 Multi-agent systems

Multi-agent systems (MAS) is a broad paradigm and/or research topic. It is a subfield of Distributed problem solving. A multi-agent system can be, in a short way, described as a group of autonomous agents that act towards their objectives in an environment to achieve a common goal [17]. The agents can be defined as independent units in an environment, forming a system. They are able to act independently, possess knowledge and communicate with each other (i.e. share the knowledge). The agents should be working towards some form of a common goal, which could be achieved either by cooperating or competing. The agents usually have a perception through which they can gain knowledge from their environment.

The basic definition of MAS suggests that they should be used to model/represent a system that is not centralized but rather distributed, where autonomous, intelligent units perform actions independently. Multi-agent systems have gained popularity in the recent years, as they offer high flexibility of modelling highly non-linear systems and offering abstraction levels that make it more natural to deal with scale and complexity in these systems [18]. It is apparent that MAS excel in modeling social environments involving humans. MAS share a lot of features with human societies, as these societies also revolve around atomic units - persons, that act independently - each person makes decisions and acts upon their own beliefs. People also interact with each other, be it communication, cooperation or competition, while working towards some goal. There are numerous real-life examples that strongly resemble this MAS definition, for example sport teams, where each player has his own role, like a defender and striker. Although all players have the same intention (i.e. win the game), their specific action differ depending on the state of their environment, each of them acting upon their own beliefs, which makes the team resemble a distributed system. From more practical perspective, MAS paradigm can be used when building complex computer networks, for example.

3.1 Agent

Agents are the fundamental building blocks of a multi-agent system. While they can have many features and characteristics specific for their use case and are not possible to generalize, there are several elementary characteristics that define them in scope of MAS [17].

Situatedness - Agents are designed so that they interact with the environment through sensors, resulting in actions using actuators. The agent should be able to directly interact with its environment using actuators.

Autonomy - Agent is able to choose its actions without other agents' interference on the network.

Inferential capability - Agent is able to work on an abstract goal specifications, identifying and utilizing relevant information it gets from observations.

Responsiveness - Agent is able to respond to a perceived condition of environment in a timely fashion.

Social behaviour - Agent must be able to interact with external sources when the need arises, e.g. cooperating and sharing knowledge.

3.2 Agent type and architecture

As has been already stated, in order to model complex applications, the agent characteristics as well as their internal control architecture will differ between use cases. In an effort to standardize MAS development, several architectures that describe how agents work have been proposed. Generally, there are three classes of agent modelling architectures that are defined based on interaction complexity with external sources:

- Reactive agents
- Deliberative agents
- Interacting agents

An overview of the class characteristics and examples of agent architectures utilizing the respective classes will be given.

3.2.1 Reactive agent architectures

Having first emerged in behaviorist psychology, the concept of reactive agents is founded by agents that make their decisions based on limited information, with simple situation-action rules. The agents usually make decisions directly based on the input from their sensors. This type of agents is mostly suited for application where agent resilience and robustness is the most important factor, instead of optimal behaviour. An example of a reactive agent architecture is the Subsumption architecture.

Subsumption architecture
This architecture described by Rodney Brooks in 1986 [19] and decomposes the agent into hierarchical levels that operate in a bottom-up fashion, meaning that bottom layers that control elementary behaviour are activated by the upper layers that define more complex action or define goals for the agent. It is important to note that the behavioral modules map sensations directly to actions and can only define what the agent does, not being able to change its desires. An example of a subsumption architecture is depicted in fig. (7) with example modules from each hierarchical level, where the bottom-level module is `Avoid Objects`, which is a needed action in order to complete the `Wander Around` action, which can be induced by the general goal on the top layer `Explore World`.

./img/Subsumption_Architecture_Abstract_Diagram.png

Figure 7: Example of a Subsumption architecture [20]

3.2.2 Deliberative agent architectures

Compared to a reactive agent, deliberative agents have a more complex structure and are closer to a human-like, rational behaviour. A deliberative agent is defined as one that possesses an explicitly represented, symbolic model of the world, and in which decisions are made via symbolic reasoning [21]. In other words, the agent maintains its internal representation of the external environment and thus is capable to plan its actions, while being in an explicit mental state which can dynamically change. The Beliefs, Desires and Intentions (BDI) architecture is the most widely known modelling approach of deliberative agents.

BDI architecture
The BDI paradigm has been used in various applications, such as simulating impacts of climate change on agricultural land use and production [22] or to improve internet network resilience by creating BDI agents that combats DDoS attacks [23]. The main idea behind this architecture is the emphasis on practical reasoning - the process of figuring out what to do. There are three logic components that characterize an agent:

Beliefs - The internal knowledge about the surrounding environment, which is being constantly updated by agent's perception.

Desires - What the agent wants to accomplish. An agent can have multiple desires, which can be hierarchically structured or have different priority.

Intentions - Intentions are formed when an agent commits to a plan in order to achieve a chosen goal. The plans are pre-defined within an agent, formally called a *plan library.* The plan that an agent has set to carry out can dynamically change based on updated beliefs or desires.

These components together define an agent's *reasoning engine* (fig. (8)), which drives the agent's (deliberative) behaviour.

This definition can be made clearer with a simple example scenario - a waiter in a restaurant.

./img/bdi-infographic.png

Figure 8: The BDI architecture schematic

The waiter's *beliefs* are the tables with customers and information about state of each table (i.e. choosing menu, waiting for meal, willing to pay etc.). The waiter's *desires* are to serve customers, for example accept order from a customer. The waiter carries out his desires by making a plan of *intentions* (e.g. go to table and ask if the customer wants a drink).

This architecture has its advantage in the fact that the functional decomposition of the system is clear and intuitive. However, with this architecture, there is a commitment-reconsideration tradeoff that needs to be optimized [24]. With too much commitment, there is a risk of agent overcommitment, where an agent might be trying to achieve a goal that is not longer valid. On the other hand, if an agent reconsiders too often, there is a risk that the agent will not achieve any goal because it will switch between intentions too quickly.

3.2.3 Hybrid approaches

Hybrid architectures try to utilize the best of both worlds of agent modelling. Purely reactive agents might lack the ability to solve complex tasks, whereas deliberative architectures are challenging to successfully implement on a concrete problem [25]. Pre-compiling a plan library for every possible scenario that can happen in environment with vast amount of complexities is simply not feasible, due to uncertainties of following effects when agents affect the environment.

The underlying concept of hybrid architectures is to structure agent functionalities into layers that interact with each other. This provides several advantages, namely *modularization*, which decomposes the agent's functionalities into distinct modules that have determined interfaces. This helps to deal with design complexity. Furthermore, having distinct layers enables them to run in parallel, thus increasing an agent's computational ability as well as reactivity [26].

Generally, amongst the most widely used hybrid architectures, a *controller* layer can be found, which handles reactive tasks therefore is also connected to the sensor readings, and is hierarchically on the lowest level. Then, there are *planning* layers that handle the logic-based, deliberative tasks and often interact with the controller layer. In-between them, there is usually a *sequencing* layer that can suppress output from the reactive layer. An example of a well-known hybrid architecture is the ${}_3$T architecture, whose abstract model can be seen on the figure (9) below.

./img/3t-arch.jpg

Figure 9: An architecture model of a ${}_3$T hybrid architecture [27]

${}_3$T architecture

This architecture builds upon a predecessor architecture called Reactive Action Packages (RAPs) [28]. A RAP is essentially a process or a description of how to complete a task using discrete steps. Note that it has got no planning abilities, i.e. its actions are only based on current perceived environment state and not on an anticipated state. When a RAP is executed, it should finish only when it satisfied its outcome or else it will produce a failure state. This ensures that the agent can self-diagnose a failure and implement some fail-safe mechanisms. Individual RAPs are queued by the interpreter, in the case of the ${}_3$T architecture it is called a *sequencer*, which is the intermediate layer between the *reactive skills* and *deliberation* layer.

The skills layer is a collection of so-called *skills*. Skills provide an interface of an agent to its environment. They are its abilities that allow the agent to transform or maintain a particular state in the environment. Each skill has got an expected input and output, which allows to route them together.

Finally, the deliberation layer is responsible for planning on a high level of abstraction, in order to make its problem space small. Routine sequences of tasks should not be specified or dealt with at this layer.

3.2.4 Conclusion

The different architecture types that were presented can each have a different application where they excel. It is also important to note that a line between reactive and deliberative agents is not necessarily strict, hence the existence of hybrid architecture types.

Choosing the optimal agent architecture will depend on the scope of agent goals, environment and action space definition. With respect to the topic of this thesis, which is to create a framework for implementing various ITS, there isn't a clear-cut choice regarding architecture selection. Although the vast majority of ITS share the same goals (see section 2), the operating environment, underlying concepts and also used technology vary substantially.

To put things into a perspective, let's consider ITS introduced in section 2. Systems such as autonomous driving algorithms require high resilience, determinism, and high performance, all in extremely dynamic conditions. This would suggest to use a *reactive* architecture. On the other hand, there are systems such as traffic network management, where highly complex problems with lots of variables are considered. Such non-linear behaviour requires complex logic and frequent re-planning, therefore more suited for *deliberative* architectures. Furthermore, when considering the aforementioned cooperative ITS, where the emphasis on information sharing and environment awareness is given, it becomes clear that *hybrid* architectures which offer both planning and reactive behaviour would be the optimal choice.

As such, when creating the framework for ITS implementation later in this thesis, the $_3$T architecture with RAP utilization will be used.

3.3 Interaction between agents

It is safe to say that an agent-based ITS system will require some form of agent interaction and thus communication between individual agents. Interacting agents are able to interact with other agents within the environment. This concept extends an agent with an interface dedicated to communication, which makes them able to directly communicate and thus cooperate in a decentralized fashion. The concept of cooperation between agents is important in the ITS modelling context, as these systems facilitate sharing information in-between drivers and also from the road traffic environment to drivers. Therefore, interaction is a strong component when considering modelling ITS and road traffic in general. However, this interface also adds to the system's complexity.

It is expected that most agents would achieve their common goal through cooperation, where some form of forced altruism is given to the agents. With that being said there could also be situations where conflicts of interest between agents occur with no clear positive outcome, i.e. zero-sum games.

One should adhere to some communication principles to facilitate cooperation between agents. As such, agents should communicate according to Gricean maxims (see the table below) [29]. This way, the communication and performance overhead is minimized and system is more stable.

Table 3: Gricean maxims

Quantity	Say not more nor less than it is required
Relevance	Stay relevant to the topic of discussion
Manner	Avoid obscurity and ambiguity
Quality	Do not give false or unsupported information

It needs to be said that the agents developed for an ITS system will most likely not be competing amongst each other, as the agent reward should be higher the more the agents cooperate. For example, when designing cooperative intersections system, the reward function for the individual intersections (i.e. agents) should take into account not only the throughput on their own intersection but also throughput of the whole system. However, that doesn't eliminate the need to negotiate. Consider a situation where there are multiple traffic light intersections, each deciding on which phases to enable based on the incoming traffic intensity. An optimal option that minimizes cost (e.g. travel time) for one agent could have a significant negative effect on other intersections. Therefore it is important to achieve an equilibrium that minimizes the overall cost.

3.3.1 Organizational decomposition

Before defining the communication protocol, which is arguably the most important aspect of agent interaction, it is important to think about the the overall system topology and inter-agent relationships. There are two possibilities when designing agent structure.

Hierarchical organization
In hierarchical organization [30], agents are organized in a tree structure, where each level has got a different level of autonomy. The flow of control is from top to bottom, i.e. agents in lower level of hierarchy conform to decisions from higher levels.

A simple form of hierarchical organization ensures that there is a low number of conflicts and the system can be operating by relatively simpler sequential processes where the control flow is straightforward, however, this also decreases the robustness of the system, because the control and autonomy not as distributed, e.g. when a failure of a single agent with at a high hierarchical level causes the whole system to fail. A subtype of this organization - a uniform hierarchy, the authority is more distributed among the agents.

This makes the system more fault tolerant and perform graceful degradation in scenarios where one ore more parts of the system fail [17]. Here, a special attention needs to be given to conflict resolution, which is not always as straightforward, as mentioned earlier.

./img/hierarchy.png

Figure 10: Hierarchical agent structure schematic [17]

Coalitions
Another useful agent organization is to organize agents into coalitions. In coalitions, a group of agents come together for a short time to increase the utility or performance of the individual agents in a group [17]. After the goal is reached or the coalition no longer becomes feasible, the coalition ceases to exist. The coalition can have a flat hierarchy, with the possibility to have one agent as the leader role for interactions outside the coalition. The use of coalitions allows for a highly dynamic system, which on the other hand increases the system's complexity.

There are more concepts for structuring a multi-agent system from organizational point of view, such as *teams* and *holons*. However, the two mentioned approaches are the most well-suited for an ITS system development, as they have been used in more works for this purpose already (e.g. [31] and [32]).

In conclusion, the usage of the aforementioned organization should ensure that the system complexity is kept as low as possible and the topology is transparent and uncluttered. The hierarchical organization can be beneficial when applied to ITS systems such as urban traffic management, where conflicts of interests need to be resolved by a capable authority. Whereas, the coalition-based organization could be applied to create virtual clusters of connected vehicles (platooning) to apply C-ITS features, e.g. dynamic navigation or GLOSA.

3.3.2 Communication between agents

As has been stated before, agent communication is a crucial component of a multi-agent system. Communication makes for inter-agent interaction beyond cues that an agent receives from the environment through its sensors, as agents can either exchange or broadcast information which could not be obtainable for the agent otherwise. This allows for deeper and more complex decision making and behaviour design - agents can act upon the received information or update their beliefs about the world and provide distributed problem solving in general.

3.3.3 Agent Communication Language

Any language, including the one used by agents in an arbitrary system, should have defined its syntax and semantics to be an effective medium. Effectively, this means that a dedicated communication interface for each agent should be defined, with a standardized way of expressing information. To satisfy these requirements, a language developed specifically for MAS has been created, called *Agent Communication Language* (ACL) [33]. This specification proposes a standard for agent communication. Most importantly, it defines so-called *communicative act* (CA) - a special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act. CAs are modelled on speech act theory. Pragmatically, CA's are performed by an agent sending a message to another agent, using a specified message format. For the index of the defined CAs and their classification, see the table (4).

Using all of the defined message types is not mandatory in order to use the ACL. However, the standard introduces ACL-compliant agent requirements that need to be fulfilled. The agent requirements are in the table (5).

Apart from the mentioned communicative acts, the ACL standard also defines message parameters, which form the structure of a message (table (6)). A sample composed message structure can be found in figure (11).

`./img/acl-message-structure.png`

Figure 11: Main structural components of ACL message [33]

It needs to be considered that there are plenty of other standard definitions that are suited for distributed system interoperability, such as W3C, CORBA, UML and many others, so the choice of the ACL needs to be argued. A lot of these applications are based on the so called CRUD set of communication primitives - Create, Read, Update and Delete. In contrast to this, ACL defines a lot of different communication primitives (i.e. communicative acts). This leads to more complex control [34]. Also, [34] states that The

ACL model naturally allows more semantic context to be included in messages. This can give applications more understandable information about unexpected events. In addition, the richer set of primitives can lead to more flexible interaction processes.

3.3.4 Conclusion

In this section, the ways how agents interact with each other were described, with respect to the application scope of this thesis, i.e. applying MAS principles to implement an ITS simulation. The most important facts are that agent agent organization needs to be considered - despite the fact that MAS are designed to be highly distributed, giving the agents hierarchical roles can often facilitate interaction between agents and can decrease system complexity while preserving functionality, especially during negotiation between agents. Agent grouping should, while also contributing to decreased system distribution, decrease computation and communication overhead.

Next, it was found that designing how agents communicate is a crucial part of MAS development that can get complex, so it is important to keep the ways of sharing information as organized as possible, which can be done by defining the language, primarily its semantics and associated syntax. The cornerstones of inter-agent communication were discussed, and a suitable framework for the thesis' use-case of setting up communication between agents was investigated and chosen. The framework of choice was the Agent Communication Language, because its set of communication primitives already assume the MAS-based use-cases, with predefined requirements, message parameters and message types while also being open for extension.

Moving forward, the following section will be dedicated to proposing the actual system that will be implemented, subject to the topic of the thesis.

Table 4: Categories of communicative acts [33]

Communicative act	**Information passing**	**Requesting information**	**Negotiation**	**Action** performing	**Error** handling
`accept-proposal`			■		
`agree`				■	
`cancel`				■	
`cfp`			■		
`confirm`	■				
`disconfirm`	■				
`failure`					■
`inform`	■				
`inform-if (macro act)`	■				
`inform-ref (macro act)`	■				
`not-understood`					■
`propose`			■		
`query-if`		■			
`query-ref`		■			
`refuse`				■	
`reject-proposal`			■		
`request`				■	
`request-when`				■	
`request-whenever`				■	
`subscribe`		■			

Table 5: The ACL-compliant agent requirements [33]

Requirement 1: Agents should send not-understood if they receive a message that they do not recognise or they are unable to process the content of the message. Agents must be prepared to receive and properly handle a not-understood
Requirement 2: An ACL compliant agent may choose to implement any subset (including all, though this is unlikely) of the predefined message types and protocols. The implementation of these messages must be correct with respect to the referenced act's semantic definition.
Requirement 3: An ACL compliant agent which uses the communicative acts whose names are defined in the specification must implement them correctly with respect to their definition.
Requirement 4: Agents may use communicative acts with other names, not defined in the specification document, and are responsible for ensuring that the receiving agent will understand the meaning of the act. However, agents should not define new acts with a meaning that matches a pre-defined standard act.
Requirement 5: An ACL compliant agent must be able to correctly generate a syntactically well formed message in the transport form that corresponds to the message it wishes to send. Symmetrically, it must be able to translate a character sequence that is well-formed in the transport syntax to the corresponding message.

Table 6: Pre-defined message parameters [33]

Message Parameter	**Meaning**
`:sender`	Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act.
`:receiver`	Denotes the identity of the intended recipient of the message.
`:content`	Denotes the content of the message; equivalently denotes the object of the action.
`:reply-with`	Introduces an expression which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously.
`:in-reply-to`	Denotes an expression that references an earlier action to which this message is a reply.
`:envelope`	Denotes an expression that provides useful information about the message as seen by the message transport service.
`:language`	Denotes the encoding scheme of the content of the action.
`:ontology`	Denotes the ontology which is used to give a meaning to the symbols in the content expression.
`:reply-by`	Denotes a time and/or date expression which indicates a guideline on the latest time by which the sending agent would like a reply.
`:protocol`	Introduces an identifier which denotes the protocol which the sending agent is employing.
`:conversation-id`	Introduces an expression which is used to identify an ongoing sequence of communicative acts which together form a conversation.

4 Proposed system

This section is dedicated to designing the system that will be used to implement ITS in the vehicle simulator software. A framework dedicated for generic MAS-based ITS system implementation will be designed, utilizing the principles and paradigms gathered in the preceding sections (2, 3). Firstly, the *micro-architecture* will be defined, i.e. the specification of the system's actors (agents) - their features, characteristics, state variables and their goals, as well as interfaces to the environment. This will form the elementary foundation that will be used to build an actual system. As a follow up, agent interaction interface will be defined. This will for example include knowledge sharing, conflict resolution and overall communication interface proposal, including shared vocabulary and communication layers definition. The interaction interface will, subsequently, lead to the *macro-architecture* proposal. The structure of the system and will be defined. These steps should lead to a full system specification that will serve as a framework to implement agent-based ITSs in an interactive vehicle simulator.

4.1 Agent/micro architecture overview

As per the previous section(s), where the individual MAS architectures have been reviewed (section 3), it was decided to utilize the hybrid *$_3$T architecture*[1] (section 3.2.3), which will offer sufficient flexibility. Such modeling flexibility is needed primarily because there won't be a single, concrete system to model, but rather a generic system that will facilitate arbitrary agent-based ITS implementation. As such, it makes sense to choose a hybrid architecture, which will ensure there will be optimal balance between robust, reactive behaviour without giving up capabilities to model complex behaviour.

Note that the architecture will be formally assume that the its implementation will be realized in an Object-Oriented Programming (OOP) paradigm. There are multiple reasons for that. Firstly, The nature of agent based systems, having their internal logic and interacting with the surroundings through pre-defined interface, corresponds to a large degree to the concepts of OOP, especially the encapsulation mechanism.

The individual layers/components will be outlined in the following section, in a bottom-up fashion.

4.1.1 The architecture layers

In this section, the individual layers of the architecture will be defined. The layers' definition will adhere to the characteristics of the $_3$T architecture. For each layer, it's purpose and relation to other layers will be described, together with technical implementation guidelines, such as configuration etc.

[1]To remind the reader of the architecture's general structure, its schematic is shown below (fig. 12).

./img/3t-arch.jpg

Figure 12: An architecture model of a $_3$T hybrid architecture [27]

Reactive skills layer & Physical modules

This layer encapsulates all the *skills* the agent is able to do. These are the most primitive types of its behaviour[2]. For example, a skill might be to slow down or to follow a vehicle in case of vehicle-based agent. Skills are usually activated one at a time, activated by the superior layers. This creates a level of abstraction, comparable to the principle found in Object Oriented Programming, where the superior unit (sequencer in this case) does not care how the given instruction will be executed. The sequencer only cares about the use-case and output of available skills.

In practice, each skill will be an individual script that will be synchronously, sequentially run. Inside the script, it will be possible to interact with the agent's interface, mainly its sensors or communication modules. The sensor and communication modules may provide an additional level of abstraction, with exposed interface which will be defined by the architecture but configurable by the user. This will mostly involve defining parameters of physical characteristics of the module, such as signal range, or error rate. Speaking of error rate, a skill will be also able to return a failure state to enable fail-safe behaviour.

This essentially means that the skills will interact with another layer, which we could call the *Physical module layer*. This will be an elementary unit, which must be bound to a specific agent type. Therefore, the assigned modules will determine which skills the agent will be able to perform.

[2]The original author of the architecture refers to them as Reactive Action Packages (RAPs) [28].

Sequencing layer

The sequencing layer is responsible for *queue management* and *execution* of the individual skills. It is the intermediate layer between the reactive skill layer and deliberative planning layer.

Apart from the trivial FIFO skills queuing, the sequencer will manage concurrent queues with different priorities, as the agent has to conform to a dynamic environment. Because indirect communication (i.e. broadcasting) will be featured in the proposed system, asynchronous skill sequencing will also be implemented using *callbacks*. Usage of this feature can be simply argued by the fact that in traffic, the state of environment is changing rapidly, thus relying on sensory feedback would often not be enough to avoid faulty behaviour. Furthermore, there is a vast number of ITS solutions that utilize the broadcaster-subscriber principle, such as CACC systems and C-ITS systems in general.

Deliberation layer

The deliberation layer (planner) synthesizes high-level goals into a partially ordered *plan*, listing tasks that the agent has to perform, in order to achieve the specified goal. For example, A vehicle's goal is to get from its initial position to its destination. The deliberation initializes a plant which sequences skills that should ensure the vehicle gets to the destination point. The deliberation layer is aware of the traffic network (i.e. the environment), but hasn't got *complete knowledge* about it. In other words, the agent knows which turns to take on the network to get to the destination, but isn't aware of every vehicle, pedestrian or other "obstacle" that could get in its way. That's why, when there is, for example, a slow vehicle that can be overtaken, the deliberation layer *replans* the tasks in order to resolve the situation.

In practice, the deliberation component in this proposed framework will contain the most amount of pre-defined logic. The deliberation layer will contain solutions to actual problems, creating *workflows* created from individual steps (skills).

The most trivial option will be to create *static* workflows, which will work under the assumption that workflow will not get interrupted or reach a fail state at any point, i.e. is expected to finish. The second option will be to create *dynamic* workflows, which will enable re-planning according to changing environment conditions. Re-planning will occur when it's triggered by one of the *exit state* of agent's skills. The triggers will happen under the following conditions: skill returning a *failure state*, callback triggered by a broadcast subscription or result of inter-agent negotiation (discussed later). This will cause the planner to re-plan by adding appropriate steps to the executing workflow to optimally adapt to the situation. Therefore, the framework will offer to define conditions under which additional steps will get added to ideally achieve a *fail-safe behaviour*. For better logical organization, the added steps (which are identified to most often execute in a certain sequence) could be grouped into *subtasks*.

To further improve the resilience of the system, the executing workflow can be configured to reach a *workflow failure state*, which will trigger a pre-determined fail-safe workflow, that is composed of entirely different steps, essentially throwing away the previous, failed workflow. This will be useful when the planner will run out of options to adapt to the situation, settling for a goal that would destabilize the system as least as possible. For example, autonomous vehicles performing a *minimum risk maneuvre* to come to a standstill on the road side when the expected driver input is not received [35].

A more detailed view on the individual components of the architecture is on the image (13) below.

./img/3T-arch-edited.png

Figure 13: An overview of the proposed 3T architecture [36] (*edited*)

4.2 Macro architecture

4.2.1 Communication protocol

As has been discussed in the previous sections, communication plays an important part in MAS, mainly because it vastly extends capabilities of interaction between agents and thus the ability to model more complex behaviour. In section (3.3.3), the ACL communication standard for MAS has been introduced and requirements for the protocol implementation have been presented. To be able to utilize the communication, requirements for messaging utility and high-level implementation details will be presented.

4.2.2 Broadcasting

The first type of communicating that will be implemented is message broadcasting. This is argued by the fact that ITS uses information broadcasting in a lot of cases. Generally, ITS services have common communication requirements [37]:

- **Periodic status exchange** - ITS services often need to know about status of other actors, such as terminals or vehicles. These periodic updates often include basic status messages such as location, ID, speed, etc.
- **Asynchronous notifications** - Messages that are used to inform about specific event, usually related to a specific service and functionality.

The framework should therefore support message broadcasting for both synchronous and asynchronous events, which should ensure that all potential use-cases are covered.

As has been mentioned above, the ability to broadcast and subscribe to said broadcast should be handled by a separate "physical module" that will be active throughout execution of individual skills, i.e. its lifetime should not depend on other skills lifetime. An exception to this would be the skills whose purpose is to *register* and *un-register* those communication modules. The broadcast and subscription will work like most such communication systems, where each agent chooses a stream to connect to and publish/consume messages from queue (with pre-defined semantics i.e. CAM, DENM). The ability to act upon a received message will be also implemented, by making a callback that makes the skill end and return.

4.2.3 Negotiation & Agreement

Negotiation between agents is important especially when wo agents run into each other, followed by the currently executing skills of the agents getting interrupted. In other words, The best course of action of one agent is not necessarily best to the other. This is more than likely to happen in environments where multiple agents share the same resource and their state is defined on the same domain, which is often the case. As such, the theory of games is formally used to model these interactions [36].

This is quite different from the principles of agent cooperation. In cooperation negotiations, the worst-case scenario improvement as opposed to working individually is net-zero. [36] defines four speech acts in agent negotiation, as seen on fig. (14). This is inherently contrary to the statement made in chapter (3.3), where it was suggested that agent cooperation should be emphasized. However, as agents are formally modelled as self-interested, it cannot be guaranteed that they won't encounter conflicts between each other. Therefore, modelling cooperation before figuring conflict resolution would inherently result in a more fragile system more prone to malfunction.

`./img/speech-acts.png`

Figure 14: Types of agent negotiation [36]

4.2.4 Conflict detection

As has been argued above, because of scope limitation of this thesis, conflict resolution as a system functionality should precede cooperation. The cooperation abilities of agents can be added later by extending the framework. This decision will theoretically lead to better system stability, before offering extended behaviour possibilities.

The first step is to have a way for agent to determine a conflict has happened. Agents wouldn't be able to resolve conflicts if they didn't know it has occurred. To comply with the $_3$T architecture topology defined above, this should be handled on the reactive skills layer. Consequentially, detected conflict will be handled through a aforementioned module that will be bound to an executing skill. If an agent is expected to run into a conflict with another agent (of the same type), if will have to be "equipped" with a dedicated module that will behave as a sensor detecting conflict with the implementation left to specific use-case. In other words, the logic for conflict detection will be left to define together will specific agent and its properties, and other modules.

This will make the conflict detection process very flexible, as agent will have the "luxury" to combine his sensory feedback with communication modules to detect conflict with another agent. This will also solve the potential issue with only one agent acknowledging there is a conflict between it and other agent(s). Namely when one agent detects a conflict through one of its modules, it can inform other agents about the conflict as a consequence and all agents will be aware of the situation.

4.2.5 Conflict resolution

Now that the conflict detection process was defined, in that case, it will be possible to build on it to define how conflicts will be resolved. Considering the findings in chapter 3.3.1, the best tool to resolve conflicts is to define hierarchy in the agent. That is, assigning each agent a hierarchical level that determines his bidding power. As the figure (14) depicts, when . In relation to the figure (14), conflict is essentially a a competition about a resource. The said resource can have many forms apart from the more obvious free space (fig. (15)), more generally, it can also be defined as a right to decide first what action to take, on the expense of the other agent(s) in the conflict. In order to avoid more complex mathematical reasoning models [36], we can determine the bidding power by one or more agent-bound resource types that each have a value assigned. This value can be either predefined (e.g. hierarchical level) or change dynamically based on agent state and the surrounding environment (e.g. relative position of the agent).

Example

To illustrate how the system would work in practice a sample scenario is presented below on figure (15), where the agents are drivers/vehicles. The scenario demonstrates how agents achieve their goals using the three layers, as well as conflict detection and resolution.

There are two vehicle agents (`A`, `B`) in the scenario, which both have three skills defined: `drive`, `avoid:object` and `wait` and `negotiate`. Both vehicle agents have a goal not to crash and keep a certain speed. To ensure a fail-safe behaviour, a dynamic workflow is used, defined as a mapping (`failed_skill`→`safe_skill`):

- `drive` → `avoid:object`
- `avoid:object` → `wait`
- `wait` → `fail_goal` (after a timeout for example)

The mapping of the `negotiate` skill is `drive` or `wait` based on a negotiation result for each involved agent.

1. Both agents do not detect any obstacles in their surroundings, so the planner initializes a plan with only the `drive` skill sequenced, which gets executed.
2. Agent (`A`) spots an obstacle in his way (an oil spill). This makes the `drive` skill fail and (`A`)'s planner has to re-plan to reach its goal (not crash). The failed skill's fail-safe skills are retrieved (`avoid:object`) and added to the new plan. The new plan is initialized with the two following skills sequenced: `avoid:oil_spill` → `drive`.
3. However, this plan also fail, as agent (`B`) detects a path conflict, consequentially informing agent (`A`) about it. They now have to negotiate to reach an agreement and resolve the conflict using auction-like bidding. The agent (`B`) wins the bid, due to being in a right lane and so agent (`A`) is forced to give way. The planner creates a new plan: `wait` → `avoid:oil_spill` → `drive`.
4. The individual skills finish successfully and the agents' goal is fulfilled.

4.3 System requirements

To sum up the topic discussed in previous chapters, requirements for framework implementation will be concluded. This will ensure that when integrating this ITS framework into existing system (i.e. IVS software), the requirements will be organized and kept as a reference, helping achieve the desired functionality. Especially when later choosing a library/framework to help build the framework.

4.4 Conclusion

In the preceding sections, the multi-agent system framework was proposed. This framework will be used to implement various suitable ITSs into an interactive vehicle simulator. The system was described both on micro and macro level. The micro level mainly addressed specification of inner architecture of individual agents, how they will achieve their goals using their own intelligence. For building independent agents that need to be flexible in terms of their capabilities, the $_3$T architecture was chosen and the individual layers of the architecture were specified in detail, specifying their responsibilities and capabilities.

Afterwards, the macro architecture was specified, addressing mainly the ways of how agents

./img/AgentPlanning.png

Figure 15: An example behaviour of vehicle-based agents using the proposed architecture

Table 7: Proposed system requirements

Requirement 1: The framework should be a multi-agent based system, supporting more actors that are able to act independently, having their own logic and able to make decisions based on their own perception of the environment.
Requirement 2: The framework should support the ${}_3$T architecture. This primarily means implementation of the three layers of logic that offer complex behaviour modelling.
Requirement 3: The framework's architecture should be modular enough to support broad spectrum of ITS implementation.
Requirement 4: The actual implementation of a particular actor's abilities should be done using elementary skill modules.
Requirement 5: An agent should be able to interact with the environment using physical modules that are bound to individual skills. The physical sensor modules should have a given interface to provide expected output and/or input.
Requirement 6: Agent should be able to communicate with others through a dedicated communication interface, with pre-defined semantics. The communication should also adhere to the ACL-compliance requirements mentioned in section 3.3.3.
Requirement 7: The supported communication modes will be both direct (messaging specific agents) and indirect (broadcast & subscription).
Requirement 8: Agent should be able to act upon received information from physical modules, i.e. sensors and communication interface. That means, being able to process received information on skill-level, and signaling to the deliberation layer appropriately.
Requirement 8: Execution of the elementary skills should be managed by the sequencing layer. The sequencing layer should provide interface to facilitate such management.
Requirement 9: Each agent should have its role determined by defining its workflows. The workflows are series of steps that should finish as soon as the agent reaches its main goal.
Requirement 10: It should be possible for an agent to adapt to the environment by dynamic re-planning. this will be achieved by offering conditional sub-tasks that will be managed by the conditional logical reasoning system in the deliberation layer.
Requirement 11: Execution of these conditional sub-tasks should be based on exit states of individual skills.
Requirement 12: The deliberation layer should be able to process all skills' exit codes that were defined.
Requirement 13: It should be possible to implement a fail-safe behaviour, which gets activated in case the specified goal cannot be reached by the agent. In such case, a specific fail-safe workflow is defined.
Requirement 14: Where applicable, agents should be able to detect conflicting intentions with other agents while executing their tasks. The conflict detection will be implemented as a module on the reactive layer.
Requirement 15: On conflict-detecting agents, they must come with a communication module, to be able to notify other about the conflict or receive such notification.
Requirement 16: Conflict-detecting agents should be either able to resolve the conflict or trigger fail-safe state.
Requirement 17: Conflict resolution should be done by bidding a pre-defined resource(s). Hierarchy level can also count as a resource.

will interact with each other. Firstly, the communication interface was defined, including how agents will be able to use it on micro-architecture level. Building upon that, the ways of how conflicts between agents are handled was proposed, utilizing communication to resolve conflicts through resource bidding.

With the micro- and macro-architecture being specified, the final requirements for the system implementation were proposed, which aim to help ensure that the software framework will be able to facilitate ITS implementation into IVS software in an organized way, with a high system resilience.

5 System implementation

With the finished system specification, it should be discussed which tools should be used to build the software framework. First off, it is important to analyze the IVS simulation system/software (i.e. the super-system) that the proposed framework will be incorporated into. An effort should be made to maximize the super-system acceptance of this system by choosing an optimal tool-set for its implementation. This, for the most part, refers to an optimal choice a programming language and a subsequent agent-based modelling library to use (if there will be any), also due to this being one of the primary tasks of this thesis. Therefore, the next section will be devoted to introduction to the IVS software.

5.1 Simulator software

The particular simulator software that is being used at the university's faculty is being developed using the *Unity* game engine, developed by Unity Technologies. The game engine was first released in 2005 and has been used to develop numerous simulators as well as other video games ever since [38]. Unity has also been used as a tool in physical product modelling, AI & machine learning and digital twins, used in many industries [39]. The main strengths of Unity are multi-platform development support, virtual reality development support, good community support (including its own asset store) and good documentation. The game engine has got its own development environment, which can be seen on figure (16) below.

The game engine's runtime is written in the C++ programming language, but the scripting API that it offers is in the C# language. Consequently, the libraries that will be used to develop the system should ideally also be .NET based to achieve maximum customization and interoperability. Potentially, the Unity's Asset store could offer MAS packages, directly utilizing Unity's API.

Figure 16: The Unity development platform UI

References

[1] European Parliament. *Road fatality statistics in the EU*. June 2021. URL: https://www.europarl.europa.eu/news/en/headlines/society/20190410STO36615/road-fatality-statistics-in-the-eu-infographic.

[2] W. Wijnen et al. *Crash cost estimatesfor European countries*. Research rep. , Deliverable 3.2 of the H2020 project SafetyCube. European Commission, 2017. URL: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b1e92ba3&appId=PPGMS.

[3] ETSI. *ITS schematic*. URL: https://www.etsi.org/technologies/automotive-intelligent-transport.

[4] *Intelligent Transport Systems - Road*. 2022. URL: https://transport.ec.europa.eu/transport-themes/intelligent-transport-systems/road_en.

[5] Jonathan Levy. *Emissions from traffic congestion may shorten lives*. 2011. URL: https://www.hsph.harvard.edu/news/hsph-in-the-news/air-pollution-traffic-levy-von-stackelberg/.

[6] Francesco Corman et al. "Centralized versus distributed systems to reschedule trains in two dispatching areas". In: 2 (2010), pp. 219–247. ISSN: 1866-749X. DOI: 10.1007/s12469-010-0032-7.

[7] Andy H.F. Chow, Rui Sha, and Shuai Li. "Centralised and decentralised signal timing optimisation approaches for network traffic control". In: *Transportation Research Procedia* 38 (2019). Journal of Transportation and Traffic Theory, pp. 222–241. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2019.05.013. URL: https://www.sciencedirect.com/science/article/pii/S2352146519300225.

[8] *ERTICO*. URL: https://ertico.com.

[9] Hannah Ritchie, Max Roser, and Pablo Rosado. "CO2 and Greenhouse Gas Emissions". In: *Our World in Data* (2020). https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions.

[10] *C-ITS: Cooperative Intelligent Transport Systems and Services*. 2022. URL: https://www.car-2-car.org/about-c-its.

[11] *C-Roads brochure*. 2021. URL: https://www.c-roads.eu/fileadmin/user_upload/media/Dokumente/C-Roads_Brochure_2021_final_2.pdf.

[12] European Commision. *Final report of the Single Platform for Open Road Testing and Pre-deployment of Cooperative, Connected and Automated and Autonomous Mobility Platform (CCAM platform)*. 2021.

[13] Anshul Saxena. *Everything You Need to Know About In-Vehicle Infotainment Systems*. URL: https://www.einfochips.com/blog/everything-you-need-to-know-about-in-vehicle-infotainment-system/.

[14] Bart van Arem, Cornelie J. G. van Driel, and Ruben Visser. "The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics". In: *IEEE Transactions on Intelligent Transportation Systems* 7.4 (Dec. 2006), pp. 429–436. DOI: 10.1109/TITS.2006.884615.

[15] Luigi Pariota et al. "Green Light Optimal Speed Advisory: a C-ITS to improve mobility and pollution". In: *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*. IEEE, June 2019. DOI: 10.1109/EEEIC.2019.8783573.

[16] Konstantinos Katsaros et al. “Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform”. In: *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, July 2011. DOI: 10.1109/IWCMC.2011.5982524.

[17] Balaji Parasumanna Gokulan and D. Srinivasan. “An Introduction to Multi-Agent Systems”. In: vol. 310. July 2010, pp. 1–27. ISBN: 978-3-642-14434-9. DOI: 10.1007/978-3-642-14435-6_1.

[18] Birgit Burmeister, Afsaneh Haddadi, and Guido Matylis. “Application of multi-agent systems in traffic and transportation”. In: *IEE Proc. Softw. Eng.* 144 (1997), pp. 51–60.

[19] R. Brooks. “A robust layered control system for a mobile robot”. In: *IEEE Journal on Robotics and Automation* 2.1 (1986), pp. 14–23. DOI: 10.1109/JRA.1986.1087032.

[20] R.A. Brooks. *Cambrian Intelligence: The Early History of the New AI*. Bradford book. MIT Press, 1999. ISBN: 9780262024686. URL: https://books.google.cz/books?id=LZa3QgAACAAJ.

[21] M. Wooldridge. *What agents aren’t: a discussion paper*. UNICOM Seminar on Agent Software, 1996. DOI: 10.1049/ic:19960648.

[22] Philippe Caillou et al. *A Simple-to-use BDI architecture for Agent-based Modeling and Simulation*. 2017. DOI: 10.1007/978-3-319-47253-9_2.

[23] Ingrid Nunes, Frederico Schardong, and Alberto Schaeffer-Filho. “BDI2DoS: An application using collaborating BDI agents to combat DDoS attacks”. In: *Journal of Network and Computer Applications* 84 (2017), pp. 14–24. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.01.035. URL: https://www.sciencedirect.com/science/article/pii/S1084804517300577.

[24] Michael Wooldridge and Simon Parsons. “Intention Reconsideration Reconsidered”. In: (1999), pp. 63–79. ISSN: 0302-9743. DOI: 10.1007/3-540-49057-4_5.

[25] Patricia Anthony et al. “Agent Architecture: An Overview”. In: *TRANSACTIONS ON SCIENCE AND TECHNOLOGY* (Jan. 2014), pp. 18–35.

[26] Jörg Müller. “Architectures and applications of intelligent agents: A survey”. In: *The Knowledge Engineering Review* 13 (Feb. 1999), pp. 353–380. DOI: 10.1017/S0269888998004020.

[27] R. Bonasso et al. “Experiences with an Architecture for Intelligent, Reactive Agents.” In: vol. 9. Jan. 1995, pp. 187–202. DOI: 10.1080/095281397147103.

[28] R. James Firby. “An Investigation into Reactive Planning in Complex Domains”. In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*. AAAI’87. Seattle, Washington: AAAI Press, 1987, pp. 202–206. ISBN: 0934613427.

[29] Yoav Shoham and Kevin Leyton-Brown. *Multiagent SystemsAlgorithmic, Game-Theoretic, and Logical Foundations*. DOI: 10.1017/cbo9780511811654.020.

[30] Ariuna Damba and Shigeyoshi Watanabe. “Hierarchical Control in a Multiagent System”. In: (2007). DOI: 10.1109/icicic.2007.334.

[31] P. G. Balaji et al. “Multi-agent System based Urban Traffic Management”. In: (2007). DOI: 10.1109/cec.2007.4424683.

[32] Michael Vijsel and John Anderson. “Coalition Formation in Multi-Agent Systems under Real-World Conditions”. In: *AAAI Workshop - Technical Report* (June 2004).

[33] Foundation for Intelligent Physical Agents. *Agent Communication Language Specification*. 2001. URL: http://www.fipa.org/specs/fipa00018/.

[34] Stefan Poslad. “Specifying Protocols for Multi-Agent Systems Interaction”. In: 2 (2007), p. 15. ISSN: 1556-4665. DOI: 10.1145/1293731.1293735.

[35] Working party on Autonomous vehicles. *Proposal for the 01 series of amendments to UN RegulationNo. 157 (Automated Lane Keeping Systems)*. Standard ECE/TRANS/WP.29/2022/59/Rev.1. United Nations, May 2022.

[36] Walter Binder. "State-of-the-art in Agent-based Services". In: (Oct. 2022).

[37] José Santa et al. "Vehicle-to-infrastructure messaging proposal based on CAM/DENM specifications". In: *2013 IFIP Wireless Days (WD)*. 2013, pp. 1–7. DOI: `10.1109/WD.2013.6686514`.

[38] Unity Technologies. *Unity*. 2022. URL: `https://unity.com/`.

[39] Unity Technologies. *Top Uses of Unity Solutions*. 2022. URL: `https://unity.com/solutions/government-aerospace`.