Models for Optimising Dynamic Urban Mobility

D2.1 Low-Carbon Traffic Management Models

WP2: Development of traffic management models

Deliverable Lead: UNIMAN
Author(s): Abdallah Namoun, Javad Akhlaghinia (UNIMAN), Johan Philips (KU LEUVEN)
Contact: abdallah.namoun@mbs.ac.uk
Manchester Business School,
Booth Street West,
Manchester, M15 6PB, UK

Contributing Partners:
TML, KU LEUVEN

Delivery Date:
15 November 2012

Dissemination Level:
Public

Version:
1.0 (final approved)

modum
Models for Optimising Dynamic Urban Mobility

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Document Information

Document Status

Deliverable Lead	UNIMAN
Type	Report
Work Package	WP2: Development of Traffic Management Models
ID	D2.1: Low Carbon traffic management models
Due Date	28/02/2013
Delivery Date	15/11/2012
Status	Final Approved

Deliverable History

Version	Date	Output	Responsible
V0.0	09/10/2012	Deliverable structure and purpose circulated to MODUM partners for comments and feedback	Abdallah Namoun
V0.1	24/10/2012	Including literature review	Abdallah Namoun
V0.2	27/10/2012	Including detailed architecture of traffic management systems	Abdallah Namoun
V0.3	31/10/2012	Circulated to partners for internal review	Abdallah Namoun
V0.4	07/10/2012	Partners' comments and feedback were addressed	Abdallah Namoun
V0.5	14/10/2012	Enhancing literature, architectures, and outlook of deliverable	Abdallah Namoun
V1.0	15/11/2012	Approval of deliverable by partners and submission of final version of D2.1	Sven Maerivoet

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Table of Contents

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

List of Figures

List of Tables

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

1 Introduction

1.1 Explanation of the Deliverable

This deliverable, D2.1, sets out the theoretical foundations for MODUM, and details the architectural design of two varying distributed traffic management systems, an agent-based and ant-based traffic simulation system. The multi-agent system (MAS for short) uses software agents to model individual multi-modal transport segments, such as road segments and rail segments. In the multi-agent system, each transport segment is represented by a software agent. Given information about the transport network and continuous real-time update of traffic situation, the multi-agent system endeavours to optimise the use of transport infrastructure to achieve a balance between traffic demands and available capacity. However, the ant-like traffic system uses lightweight agents to model the travellers' behaviour. In the ant-like system each vehicle is represented by an ant. The ant-like system endeavours to optimise vehicles' driving behaviour.

The multi-agent system unique and innovative aspects include:

- Fitting the distributed, dynamic and evolving nature of traffic.
- Calculating CO_2-efficient travel routes in a *distributed manner*.
- Representing each physical transport segment by *a unique and dedicated agent*.
- Taking into account *aggregate properties of transport infrastructure* such as traffic flow rate and density of a road segment. As such the simulation is not microscopic, but instead macroscopic.
- Calculating (combine) *multi-modal travel routes* using a bidding algorithm (Section 3.3).

The ant-based system unique and innovative aspects include:

- Current and past traffic situation (track and trace).
- Predicted traffic situation accounting for user intentions. For instance, accounting for user intentions allows visualizing to what extent the traffic participants have managed to coordinate cooperatively before intervening.
- On-line searchable solutions space. This allows users to find and evaluate alternatives, accounting for the predictions.

1.2 Purpose and Scope

Deliverable D2.1 aims to describe the architectural design for two primary traffic management systems of MODUM: a multi-agent system and an ant-based system. These architectures will be used as the foundation for future implementation and development efforts within MODUM, and will form the backbone of MODUM systems. D2.1 focuses on the internal structure and behaviour of the two distributed systems. It also reviews relevant theories, models, and concepts to inform the design decisions made about the inner workings of the two architectures. Moreover D2.1 outlines a future implementation timetable.

1.3 Structure of Document

The remainder of this deliverable is structured as follows. Section 2 reviews relevant literature concerning, multi-agent systems, ant-based systems, traffic flow models and

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

theories, and traffic management simulations. Section 3 describes the architecture of the multi-agent traffic management simulation system, along with its class diagrams, and presents initial implementation results and algorithms. Section 4 describes the architecture of the ant-based traffic management simulation system, and justifies the architectural decisions and choice of implementation language. Each architecture section concludes with an implementation plan outlining upcoming development actions along with anticipated deadlines.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

2 Literature Review

The following section of deliverable 2.1 reviews three main research threads related to the field of agent-oriented technologies, ant-based technologies, and traffic control and management, primarily:

1. Multi-agent systems
2. Traffic flow concepts and theories
3. Agent-based models for traffic management simulations

2.1 Multi-agent Systems

This literature review starts by defining the underlying concepts of a multi-agent system, and argues for its advantages. The field of multi-agent systems is viewed as a relatively new research trend and a promising breakthrough in software development, and is founded on top of various mature disciplines mainly artificial intelligence, distributed computing, sociology and philosophy (Jennings and Wooldridge, 1998).

In simple terms, multi-agent systems are defined as systems composed of a collection of interacting agents within a host environment aimed at solving complex problems that are beyond individual systems. It is a requirement for agents to be embedded in an environment which facilitates interaction and communication among agents. Multi-agent systems are advantageous as they are able to solve computational problems that may be too large for a centralised system or to improve system performance and reliability. Other benefits include interconnection of multiple legacy systems using agent wrappers and efficient coordination of multiple distributed sources of information. Interacting agents could act on behalf of software systems as well as humans / human teams. Multi-agent systems provide greater benefits when distributed over a network, allowing agents to exploit the capabilities (e.g. computational resources) of the environments where they live.

(Woolbridge, 2007) defines an agent as "a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives". As such, an essential notion of agents is the ability to perform autonomous actions without the need to receive orders from its users or without intervention from internal or external environments and systems. Typically an agent perceives the environment through sensors, reasons the information through its reasoning engine and acts upon the environment through actuators. Such an interaction is continuous. The process of mapping perceptions to actions to satisfy the goals is called behaviour. This is the typical lifecycle of agent behaviour: perceives, decides and acts. In summary, agents exhibit additional properties that distinguish them from regular computer systems:

- **Reactivity**: agents can perceive the status and changes within an environment where they exist and react to these changes in a timely manner to fulfil their goals.
- **Proactiveness**: in pursuit of their goals and the goals of the environment, agents show goal-oriented behaviour.
- **Socialability**: agents interact with each other or other non-agent systems to fulfil their design goals and the goals of the system. In the agent context, being "sociable" refers

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

to the ability to communicate, cooperate or compete to fulfil own goals or common system goals.

Among the most traditional and common ways to model and program the behaviour of an agent is the belief-desire-intention (BDI) model (Rao and Georgeff, 1995). Agents developed on top of this model possess and implement a number of mental attitudes: a set of beliefs representing their knowledge of the world state, a set of desires representing their goals or system goals and a set of intentions representing plans for achieving the desires. The BDI architecture does have a number of limitations (Michael et al, 1999); for example it does not explicitly support learning, it does not specify mechanisms for interactions between agents and it does not have an explicit representation of goals. Other alternate models for programming the rational behaviour of agents include the Soar model (Laird et al, 1987) and Markov decision process-based model (Bellman, 1957). The Soar model is based on operators (i.e. commitments) and states where operators are selected based on pre-conditions and current state of agent. A Markov decision process models the decision making process in situations where the outcomes are partially random. In such model agents hold a probability distribution of the potential states, and take actions which lead the environment to a state with a particular probability.

Agents live in an environment whose characteristics are quite crucial for the success or failure of agents. (Russell and Norving, 2003) classified agent environments into the following types:

- **Accessible / inaccessible**: this reflects the ability of the agents to obtain a complete, accurate, and up to date state of the environment. With this level of information about the environment, agents are expected to make better decisions.
- **Deterministic / non-deterministic:** an environment is deterministic if any single agents' action has only one guaranteed result in the environment. In such environment there is no element of uncertainty about the outcomes of agents' actions.
- **Static / dynamic:** an environment is considered dynamic if it changes beyond agents' control. In contract a static environment remains unchanged without taking into account the effect of agents' actions.
- **Discrete / continuous:** an environment is considered discrete if the number of actions to be performed by the agents is limited.

In contrast to the object-oriented programming paradigm which uses objects, agent-oriented programming emerged at the beginning of the 90s and uses agents and their messaging capabilities as its centrepieces to create software (Shoham, 1990). Following Shoham's programming paradigm, various frameworks, platforms and languages emerged to implement its features and properties. A complete survey of available agent-based modelling and simulation platforms and toolkits, including ABLE, AnyLogic, JADE and MASON, is provided by Nikolai and Madey (2009). The authors classified these toolkits based on five characteristics: the implementation language used for developing and running the simulation, operating system for running the toolkit, licensing policy of the toolkit, intended domain of the toolkit and level of user support provided.

Various models have been proposed to facilitate the development of multi-agent systems in a systematic manner. (Burmeister, 1996) suggested three different models for building multi-

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

agent systems: agent model, cooperation model and organisational model. The agent model describes the agents' architecture and their internal structure using mental notions (i.e. beliefs, desires, plans). The cooperation model describes the interaction and cooperation aspects of the agents. The organisational model describes the relationships between agents and agent types such as inheritance relations and role-based relations. For multi-agent systems founded on the belief-desire-intention architecture, Kinny et al. (1996) discussed the agent system from two perspectives: an external viewpoint where the system is divided into agents and the interactions among these agents through an agent class hierarchy, and an internal viewpoint where the system focuses on modelling the agents' mental notions (beliefs, desires and intentions). Other methods and methodologies for building agent-based systems, such as the multi-agent scenario based method and agent-oriented methodology for enterprise modelling, are thoroughly reviewed by (Iglesias et al, 1998).

Other prominent and widely-recognised agent-oriented methodologies include **Gaia** which consists of an analysis phase and design phase, **MaSE** (Multiagent Systems Engineering) which consists of an analysis stage, design stage, assembling agent classes stage and system design stage, **Prometheus**, which targets non-experts, consists of a system specification stage, architectural design stage and detailed stage, **Tropos** consists of an early requirements stage, late requirements stage, architectural design stage, detailed design stage and implementation stage. Dam and Winikoff (2004) developed a 60-question comparison framework focusing on agency concepts, modelling language and process and pragmatics of the methodology to evaluate the aforementioned agent-oriented methodologies.

Currently multi-agent systems are widely applied in a multitude of everyday domains including commercial, governmental, military, industrial and research fields (Intelligent Software Agents, 2010). Real-life examples include the use of multi-agent systems in email filters, air traffic control applications and financial management applications. Multi-agent systems are also used for simulation and optimisation problems such as traffic management and optimisation (Burmeister et al, 1997; Chen and Cheng, 2010), the core topic of MODUM project.

A particular class of multi-agent systems are ant algorithms. Ant algorithms use artificial stigmergy as a means for coordinating the behaviour of several agents (Theraulaz, 1999). In ant colony engineering food foraging behaviour in ant colonies is the source of inspiration for the design of the emergent generation of short-term forecasts.
The main achievement is that individual ants are not exposed to the complexity and dynamics of the situation. Instead, the environment is incorporated into the solution and allows the overall system to cope with its complexity. None of the ants need a mental map of the environment, this in contrast to typical BDI architectures (Holvoet, 2006; Rao and Georgeff, 1995). Evaporation and refreshing the pheromone trails allows the system to cope with a dynamic environment.

Ant-colony engineering, i.e. Delegate MAS applications (Holvoet, 2006), has proved its value in various applications and domains. Also in the traffic routing domain biologically inspired algorithms and virtual pheromones are often used (Tatomir, 2009; Ando, 2006). However, most approaches either focus on solving the routing problem for one individual vehicle or focus solely on the self-organization of traffic by using pheromones to stochastically guide vehicles.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

More similar to the ant-based model proposed in the MODUM project are (Tatomir, 2009; Ando, 2006; Claes 2012; Di Caro, 1998). These approaches explicitly use ant coordination in order to model travel time and congestion.

2.2 Traffic Flow Concepts and Theories

In this section we discuss the widely-recognised traffic flow theories and their fundamental concepts as it is quite crucial to understand these before developing any traffic control simulation. Theories as such aim to develop an optimal transport network with well-organised flow and reduced congestions by considering the movement and interaction between vehicles, drivers, and traffic network infrastructure. So the endeavour is to create an efficient traffic network which maximises the use of available infrastructure to eliminate traffic jams, shorten travel journeys, reduce transport carbon emissions and provide a positive and safe commuting experience for commuters.

Traffic happens in space and time, representing a spatiotemporal relationship. A space-time diagram, where distance is plotted on the vertical axis and time is plotted on the horizontal axis, allows us to graph the flow of vehicles over time. Such a diagram is beneficial for depicting and understanding the characteristics of traffic flow for roads over a period of time.

A number of traffic-related variables are often utilised to model traffic simulations and traffic flow of vehicles: vehicle speed, vehicle density, and flow rate. These quantities are defined as follows (Kerner, 2009; Maerivoet, 2006; Mannering et al, 2005; Lieu, 1999):

- **Speed (v)**: speed of vehicles represents the distance traversed per unit time by the vehicle; its unit is expressed in kilometre (or mile) per hour (km-m/h). Due to the variable nature of speed of vehicles over time and difficulty to keep track of every vehicle on the road network, average speed over a period of time or a period of space is calculated and used instead.

  v = d / t;

  There are two types of speed, time-mean speed and space-mean speed. Time-mean speed (also known as spot speed) is calculated by averaging the observed speeds of vehicles **passing a reference point**, whilst space-mean speed is calculated by averaging the observed speeds of vehicles **over a length of road**. Usually space-mean speed is found to be approximately 2% less than time-mean speed.

- **Density (concentration - k):** density defines the number of vehicles (n) per unit distance (d) at a particular instant; its unit is expressed in vehicles per kilometre (or mile) (vehs / km-m).

  k = n / d;

- **Flow rate (q):** traffic flow defines the number of vehicles passing through a reference point per unit time; its unit is expressed in vehicles per hour (vehs / h).

  q = n / t;

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Other useful traffic stream properties include **time headway** (in seconds) which defines the time delay between any two successive vehicles as their front bumpers pass a reference point, and **space headway** which defines the distance between any two successive vehicles.

Free-flow traffic: traffic is said to be freely-flowing when there is a positive correlation between flow rate (q) and vehicle density (k), until reaching an optimal flow rate alongside a critical vehicle density. After this point traffic starts to congest.

Congested traffic: in a traffic congestion case, the speed of vehicles is lower than the lowest vehicle speed observed in a free flow condition as a result of increased vehicle density. Congested traffic is characterised by three major features: slower speeds, extended journey times and increased queuing of vehicles. It is caused when the capacity of traffic infrastructure (e.g. roads) at certain points or time is smaller than the volume of traffic (i.e. number of vehicles). This is often exacerbated by other factors such as traffic incidents, bad weather conditions and road works. Traffic congestion has a number of repercussions such as trip delays, difficulty in forecasting travel times, and increased use of fuel and CO_2 emissions.

Simulation models describing traffic flow are categorised into three primary categories, microscopic, macroscopic, and mesoscopic (Hoogendoorn and Bovy, 2001; Maerivoet, 2006). Microscopic models describe the following behaviour of individual vehicles as a function of the behaviour of the leading vehicle; it thus models microscopic properties of single vehicles. Macroscopic models, however, study and establish a mathematical relationship among traffic flow characteristics including vehicle speed, vehicle density and flow rate. The easiest of these relationships is the fundamental relationship as propose by Greenshields (1935), where: **Flow rate = speed * density (q = v * k;).** In macroscopic models individual vehicles activities, e.g. lane-change, are not explicitly represented**.** Mesoscopic models present an intermediate solution between microscopic and macroscopic models, and simulate individual vehicles using aggregate macroscopic relationships. Mesoscopic models thus describe traffic flows at medium level of detail.

Several research studies attempted to tune the fundamental relationship function by considering the relationship between each pair. Car-following models, or microscopic models, study the characteristics of a single vehicle which adapts its behaviour according to the leading vehicle in the same lane. In other words, car-following models investigate how vehicles follow one another on roads considering their position and velocities. Examples of car following models (Brackstone and McDonald, 1999) in the literature include stimulus-response models (Gazis el al, 1961) where a driver responds to stimuli according to the formulae: response = sensitivity * stimulus, safety-distance models (Gipps, 1981) where a vehicle can accelerate to its desired speed but should be able to safely stop in case the leading vehicle stops suddenly to avoid collision, Newell models (Newell, 2002) where a vehicle keeps a minimum distance and time to the leading vehicle, and optimal velocity models (Bando, 1994) where the vehicle maintains maximum speed with sufficient distance to the leading vehicles and attempts to reach the optimal velocity, where acceleration represents the difference between current velocity and optimal velocity. A number of researchers evaluated the effectiveness of these models in creating traffic simulations that are comparable to complex real traffic data (Olstam and Tapani, 2004; Ranjitkar et al, 2005; Zheng et al, 2012). Other car following models are discussed in detail elsewhere (Hoogendoorn and Bovy, 2001; Marrivoet, 2006)

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

The first macroscopic traffic flow model was introduced by Lighthill and Whitham (1955) and Richards (1956) who proposed that traffic streams are comparable to fluid streams in long rivers. It is also known as the LWR model named after the authors who first described it. This first order traffic model describes the flow of compressible fluids. (Payne, 1971) modified the first order traffic model to overcome its shortcomings and improve modelling performance. Other researchers suggested macroscopic models for describing traffic flow based on the kinetic theory of gases (Newell, 1995; Bellomo et al, 2002). Models based on kinetic theory of gases compare traffic flow to ratified gases flow. Darbha et al. (2008) reviewed and critiqued in detail some of these models.

Examples of mesoscopic models include headway distribution models, cluster models, and gas-continuum models (Hoogendoorn and Bovy, 2001). Headway distribution models describe the distribution of headways of individual vehicles where a headway is the time difference between two successive vehicles passing through a reference point. Cluster models describe clusters of vehicles which share a particular property using specific aspects such as the size of cluster and velocity of a cluster. Gas-continuum models describe the dynamic changes of vehicles' velocity distribution functions in traffic steams.

2.3 Agent-based Models for Traffic Management Simulations

A number of research works attempted to address traffic management and control by creating specialised agent-based traffic simulations and systems. The application of multi-agent systems to the management and control of traffic is reasonably justified. Naturally traffic is distributed geographically, highly dynamic, and autonomous. The facts are simple, the world's population is growing fast and so are the everyday economy and leisure activities, all resulting in an overwhelming increase in the use of traffic infrastructures in all of its forms, e.g. roads, rail networks, cycle and pedestrian routes. These factors alongside the limited operational capacity of traffic segments necessitate a carefully-managed traffic network to ensure smooth fluidity and lower levels of carbon emissions. Indeed multi-agent systems can be used to plan, manage and optimise traffic by exploiting and integrating various sources of traffic information, facilitating collaboration between differing types of traffic infrastructure, and enabling autonomous behaviour of traffic subsystems. Agent-based simulation models are applied to road, air and rail traffic management (Burmeister et al, 1997).

Agent-based platforms for road traffic management and control include TRACK-R (Garcia-Serrano et al, 2003), MAS incident manager (Tomas and Garcia, 2005), and Mobile-C (Chen et al, 2009). All of these platforms are FIPA (IEEE Foundation for Intelligent Physical Agents) compliant (FIPA, 2002). TRACK-R recommends traffic routes for drivers where each agent is responsible about a geographical area. MAS incident manager uses available road traffic information to empower a road operator to manage a meteorological incident within non urban areas. Mobile-C integrates mobile agent technology with multi-agent systems to enhance the ability to detect and handle uncertainties in dynamic and distributed traffic environments. Mobile-C supports both stationary and mobile agents.

A number of researchers developed interesting agent applications that try to tackle traffic congestion problems, incidents and delays. (Wang, 2005) developed aDAPTS, a three level

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

multi-agent system for real world urban traffic control where agents perform three main tasks: organise, coordinate and execute. aDAPTS integrates the concept of mobile agent technology to empower agents to migrate between traffic centres and devices. (Roozemond, 1999) created a reactive agent based system which adapts to changing conditions in the traffic environment according to internal rules. This agent-based system consists of intersection traffic signalling agents, road segment agents and authority agents. (Weyns et al, 2007) created a delegate multi-agent system for anticipatory vehicle routing to avoid congestion problems where agents, representing single vehicles, explore alternative routes and declare their intentions in the network. (Srinivasan and Chory, 2006) proposed a multi-agent system for controlling traffic signals, where each agent is responsible about a single traffic signal of an intersection in the traffic network. Agents belonging to the same proximal zone cooperate to make a group decision.

In respect to agent-based traffic modelling and simulations, traffic entities (e.g. vehicles, signal lights, road segments etc) are modelled as agents which cooperate with each other to optimise traffic or overcome traffic problems such as congestions and route guidance. (Burmeister et al, 1997) suggested a five-module architecture for modelling a driver's behaviour in traffic simulations: sensor modules for sensing the environment, actuators modules for executing the behaviour, motivation modules for modelling agents goals and preferences, communication modules for communication between agents, and cognition modules for controlling agents activities. (Bazzan et al , 1999) suggested modelling drivers' mental attitudes using the BDI (beliefs-desires-intentions) architecture. Other research efforts to model driver behaviour include using BDI agents and agent-based framework to evaluate driving decision making behaviour (Rossetti et al, 2000), using results of a driving behaviour survey of congested areas (Dai, 2002), and using a two-layer architecture to investigate the influence of traffic updates on traffic systems and drivers' reactions (Wahle et al, 2002). (Meignan et al, 2007) developed a multi-agent traffic simulation, consisting of buses, travellers, and road traffic, for visualising and evaluating bus networks. (Kukla et al, 2001) described a microscopic model, PEDFLOW, which models the flow and movement of pedestrians in urban environments. Further agent-based systems and traffic simulation models are discussed in detail by (Chen and Cheng, 2010).

(Gibaud et al, 2011) proposed a fully-distributed self-organised approach for improving traffic flows to overcome the limitations of existing centralised and semi-distributed approaches that use a traffic information centre to measure, aggregate and diffuse traffic updates. The 'FORESEE' approach facilitates communication between vehicles which contain agents, each acting as a driving assistant to estimate the condition of surrounding traffic. Traffic information between agents is exchanged using wireless communication, and is used to optimise routes according to personal preferences, thus enabling each agent to acquire a view of traffic situation. The model employs a redundant information-reducing strategy to minimise the use of the communication medium without worsening the quality of traffic.

(Artimy 2007) identified traffic jams by estimating local traffic density relying only on the vehicle's mobility pattern. The algorithm for detecting traffic slow-downs uses the speed of vehicles-density relationship, is location-independent and does not require exchange of information between vehicles or with a central traffic information system. Instead, each vehicle uses fractions of stopped time to recognise free-flow and congested traffic situations surrounding the vehicle.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

3 MODUM Agent-based Traffic Management Model

3.1 Multi-Agent System Architecture

The goal of our multi-agent system is to model the transport infrastructure and optimise its use for a more efficient traffic flow, minimised congestion and less CO_2 emissions. By transport infrastructure we particularly refer to road segments, rail segments, bus routes, cycle paths and pedestrian routes. Our architecture makes the assumption that each road or route is composed of multiple adjacent segments, each of which is represented by a single transport agent. For instance, a road agent would represent a segment of a road and reflect the properties of this segment (e.g. length, flow rate, density ... etc). As such our definition strictly excludes dealing with intersections, roundabout and traffic signals directly. Therefore our multi-agent system does not simulate these as physical entities as the case for segments, but rather treat them as properties that contribute to defining attributes of transport segments. For instance traffic signals within a road segment regulate traffic and control traffic flow rate for that segment. In a similar manner, intersections and roundabouts control traffic flow rate to relevant segments.

We rely on static properties of traffic network segments, such as length and capacity, coupled with dynamic traffic updates obtained from live traffic sensors, such as flow rate and average speed, to optimise traffic streams. In addition we take into account user preferences and choice, such as locations of travel and time of travel, to propose personalised travel routes. Another advantage brought by our multi-agent system is its ability to combine multi-modal traffic information to recommend a multimodal route which combines various means of transport. It is worthwhile to point out that this first traffic management simulation does not aim to model and optimise dynamic driving behaviour. Instead this will be handled by the ant-based traffic management system as discussed in Section 4.

Our multi-agent system is composed of two primary units:
An environment: acts as the host for the agents where communication and interaction occur. The environment facilitates message exchanges between the differing types of agents, negotiates and resolves conflicts among environment agents.
Agents: there exist three main types of agents in the environment as follows:

- Transport (infrastructure) agents
- User agent
- Sensor agent

Transport agents: this type of agents represent segments of the transport network and include:

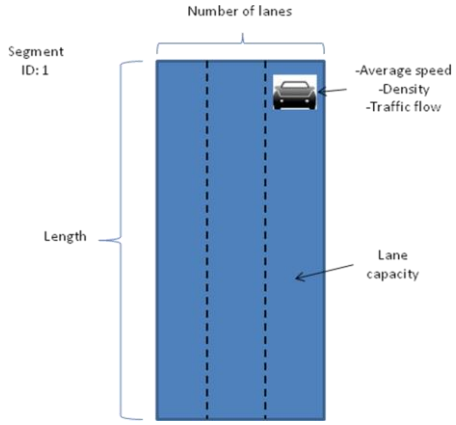
- Road agents: represent segments of the road, where a segment is the distance between any two adjacent road intersections. It is important to note here that some segments of the road might have restrictions on the time of use as in the case of bus routes which are part of the road network but may be used by cars, e.g. after 7 pm or during weekends.
- Bus agents: represent segments of the road which can be used for bus movement, where a segment is the distance between any two adjacent bus stops.
- Rail agents: represent segments of the rail way, where a segment is the distance between any two adjacent rail stations.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

- Bicycle agents: represent segments of the cycling route and road network where cycling is allowed, where a segment is the distance between any two adjacent cycle intersections.
- Pedestrian agents: represent segments of the pedestrian path where a segment is the distance between any two adjacent path intersections.

A road agent represents a segment of the road network as depicted in Figure 1. Properties of such a road segment include static infrastructure information: a segment ID, length of segment, number of lanes, segment capacity and dynamic traffic data: average speed of cars, density and traffic flow. In the road network there maybe some road segments which contain lanes dedicated for buses. During times restricting road use to buses only, designated lanes can be marked as 'unavailable'.

Figure 1 Road Segment



A bus agent represents the distance between two successive bus stops which is part of the road network. Thus a bus agent may inherit some properties of the road segments it overlaps with. Depending on the length, a bus segment could overlap with more than 1 road segment. It will have properties such as: segment ID, density, length of bus segment, average speed of bus, and flow rate. In addition a bus segment will hold additional information in the form of buses passing through it, current capacity and load of buses, and the timetable for these buses.

A rail agent models the distance between two adjacent rail stations and will maintain information about segment ID, length of segment, and current capacity. Other properties such as average speed and traffic flow are not relevant to the rail agent.

In respect to bicycle routes and pedestrian paths, the designated agents will essentially model static transport information such as ID and length. Dynamic traffic stream properties such as average speed and traffic flow are not relevant to these types of agents.

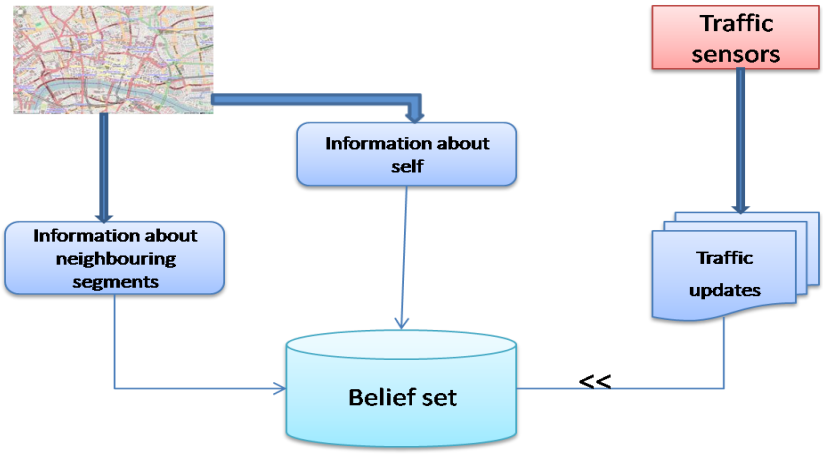
Since MODUM aims to propose multi-modal route guidance to travellers to reduce journey times and CO_2 emissions, different transport maps (e.g. road map, rail map, bus map, cycling map) will need to be linked together to exploit the advantages offered by each network and distribute the transport demands and requests. This information can be encapsulated in the belief set of each agent where a road agent, for instance, should know which (e.g. agent ID) and how many bus stops (i.e. bus agents) it overlaps with in the

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

transport network, rail stations (i.e. rail agents) and so on. The knowledge within the belief set empowers our multi-agent system to propose route guidance combining various modes of transport.

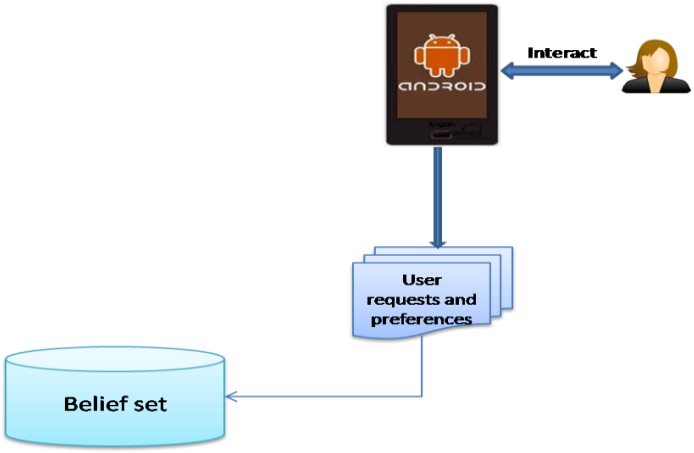
Each transport agent is unique and represents only one segment of the transport network. A transport agent holds some knowledge about itself (as shown in Figure 1), its neighbouring network agents regardless of their type, and external environment (e.g. IDs, real-time traffic information). Figure 2 depicts the sources of transport information that constitute the belief set of each transport agent. Information about self, neighbouring transport agents, and real-time traffic news are collected by the sensor agent. Whilst the transport agent holds a microscopic view of the traffic situation, the sensor agent holds a holistic view of the traffic situation. In this respect each transport agent is concerned about acquiring information that relates to identity or adjacent neighbours as depicted in Figure 2.

Figure 2 Transport Agent Belief Set



User agent: the primary function of this agent is to coordinate and manage the interaction between MODUM app users and the MODUM traffic management system. In this respect it captures user requests and preferences as text input, communicates such information to the transport agents and environment, and returns combined travel routes to the user. The best route in our case is the most CO_2 efficient travel route and could combine different modes of transport. This route is then visualised in a suitable form on the user's Android-based device.

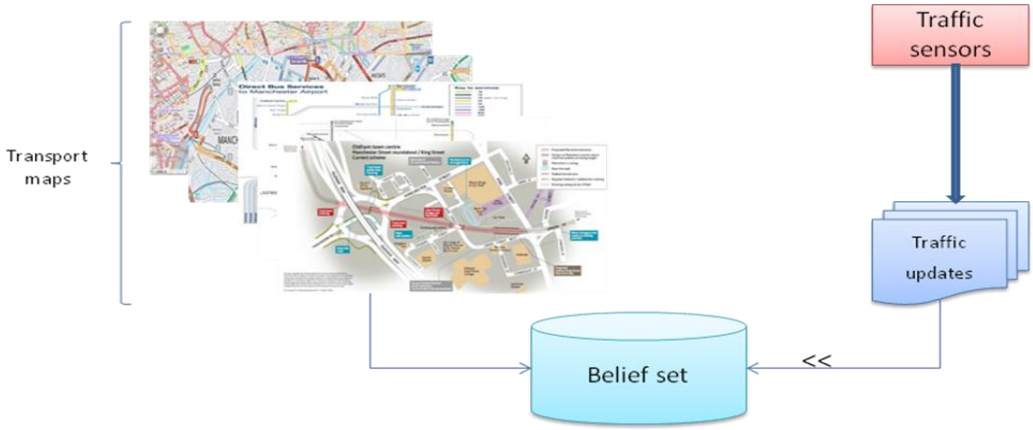
Figure 3 User Agent Belief Set



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Sensor agent: the primary function of this agent is to sense the environment for static information and dynamic traffic updates. Static information describes static properties of transport network such as the length of segments, number of lanes, capacity and timetable, whilst dynamic traffic updates describe real-time traffic data such as flow rate, density and vehicle average speed. The sensor agent reads map information once at the beginning of the simulation and checks for traffic updates on a regular basis, e.g. every 2 minutes. Such interval will be calibrated and experimented with during the traffic management simulation to reflect the true status of traffic. Map information and traffic updates are directly obtained from the traffic model simulation of WP3 as depicted in Figure 4.

Figure 4 Sensor Agent Belief Set

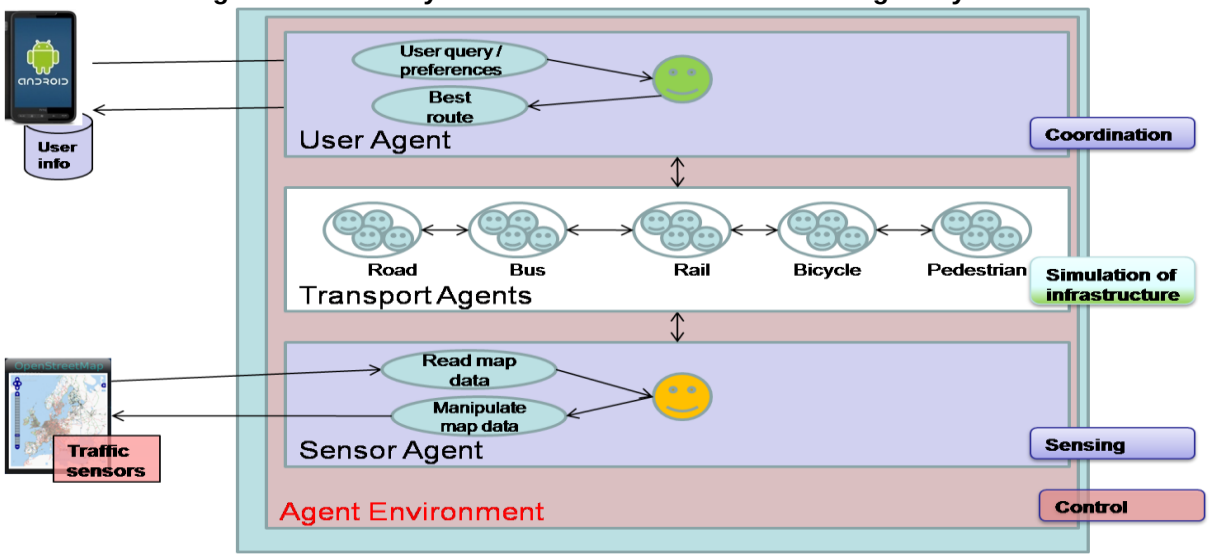


In the context of our MODUM project, it is vital for the MAS architecture to hold a symbolic representation of the transport map (e.g. information about segments and neighbours) to enable planning and to respond to changes and stimuli (e.g. traffic flow) without complex reasoning. These characteristics make our architecture hybrid (Wooldridge, 2009): deliberative and reactive.

We propose a three-layer architecture for our multi-agent traffic management system: a coordination, simulation and sensing layer (Figure 5). The coordination layer contains a user agent that coordinates user requests and delivers route guidance to users of MODUM app. The simulation layer contains transport agents which model and simulate the different modes of transport network infrastructure and try to optimise its use by achieving a balance between its capacity and current traffic demands. When calculating combined travel routes various types of infrastructure agents will need to communicate to each other. The sensing layer contains a sensor agent which updates the transport agents with information related to transport maps and real-time traffic data.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Figure 5 A three-layer architecture for MODUM multi-agent system



The three layers are contained within an agent environment where the agents co-exist. The agent environment controls all types of agents, facilitates communication between the three layers and between agents of the same type (e.g. road agents and bus agents), and resolves any conflicts among agents.

Applying the BDI (belief-desire-intention) model to our MAS architecture, we list the agent types and for each we define their goals, beliefs and capabilities. The goals signify the high level motivations of each agent, beliefs the knowledge each agent maintains about the state of the world, capability the functions each agent is capable of executing.

Table 1 Goals, Beliefs, and Capabilities of MAS

Agent Type	Goal (s) / Desire (s)	Belief (s)	Capabilities
Sensor	-Read static map data -Retrieve traffic updates	-Map information -Updates (timetables, traffic flow, CO_2 emission)	-Read maps -Collect traffic updates
User	-Interact with user -Capture user request -Delivers most efficient travel route	-Source and destination -Travel time -User preferences (e.g. driving ability, age, ownership of car)	-Capture user queries -Notify user of travel route
Transport	-Generate representations of maps and simulate the infrastructure	-Transport map segments -Neighbours -Traffic updates	-Execute plan -Communicate with other transport agents -Cooperate with other agents -Update beliefs

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

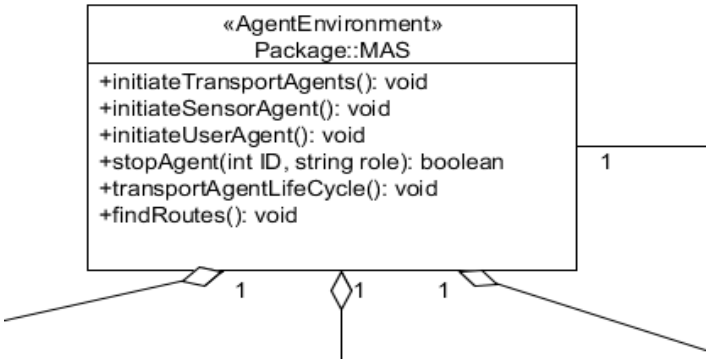
3.2 UML Diagrams

3.2.1 Class Diagrams

We use the unified modelling language (UML) (Larman, 2004) to describe the inner structure and behaviour of our multi-agent system by featuring its classes, attributes, methods and relationships between the classes. In summary the multi-agent system contains 6 main classes: Agent Environment, User Agent, Sensor Agent, Transport Agent Interface (and implementation classes), Calculate Routes, Message.

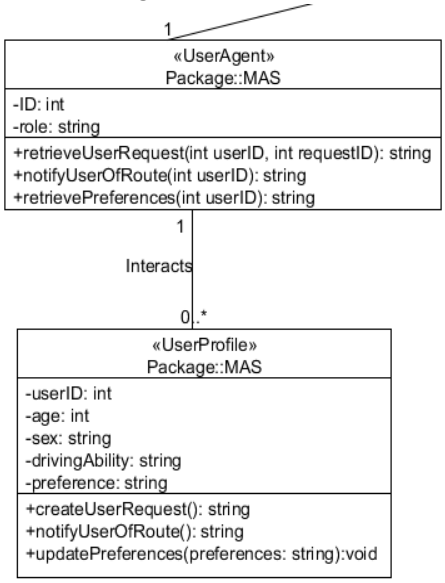
Agent Environment class: this class hosts all types of agents, facilitates communication and manages the lifecycle of the multi-agent system. It has one user agent, one sensor agent and (1 to many) transport agents depending on the transport network being simulated.

Figure 6 Agent Environment Class



User Agent class: this class enables retrieving user requests (e.g. source and destination) and commuting preferences, and notifying users of most efficient routes. This class interacts directly with the user profile class to coordinate user communication, in the form of requests and responses, with the environment.

Figure 7 User Agent and User Profile Classes

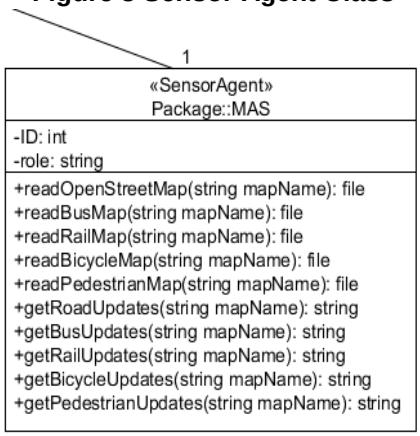


This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

User Profile class: this class enables creating user profiles for each user of MODUM multi-agent system. Each user profile has an ID and encapsulates information about its user along with commuting preferences.

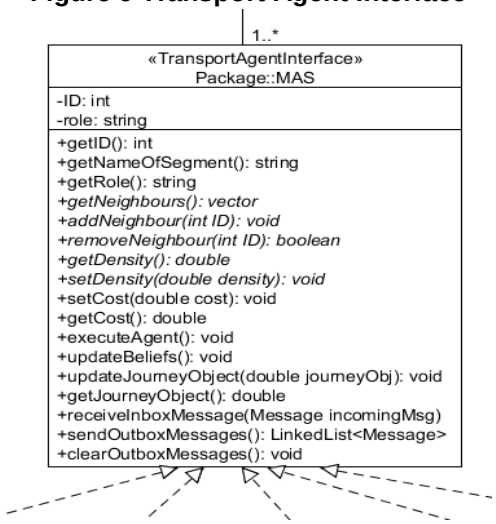
Sensor Agent class: this class enables reading transport map information and continuously retrieving traffic updates from road, bus, rail, cycle and pedestrian infrastructures.

Figure 8 Sensor Agent Class



Transport Agent Interface class: this is the interface for specifying the common attributes and methods of the transport agents to be implemented. The agent environment will have 1 to many of these transport agents.

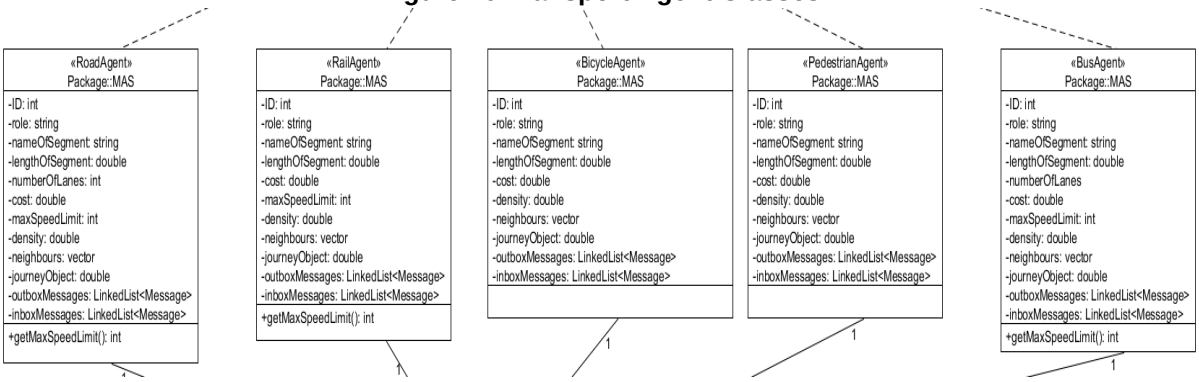
Figure 9 Transport Agent Interface



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

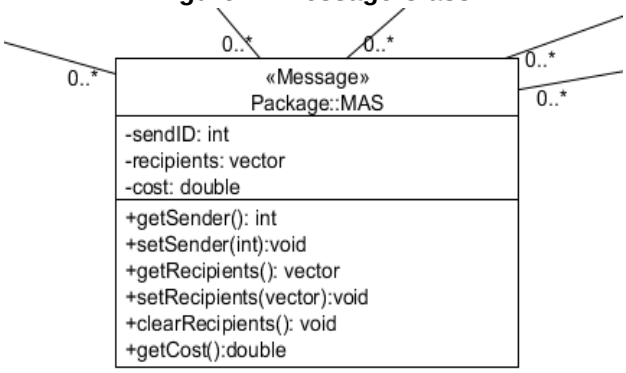
Transport Agent classes (Road Agent, Bus Agent Rail Agent, Bicycle Agent, Pedestrian Agent): these classes implement the Transport Agent Interface class and define the behaviour of the transport agents. Transport agents talk to each other using messages.

Figure 10 Transport Agent Classes



Message class: this class enables transport agents to communicate among each other, where each message has a sender, a number of recipients, and a message (i.e. cost in our case). Each transport agent can send 0 or more messages to other agents.

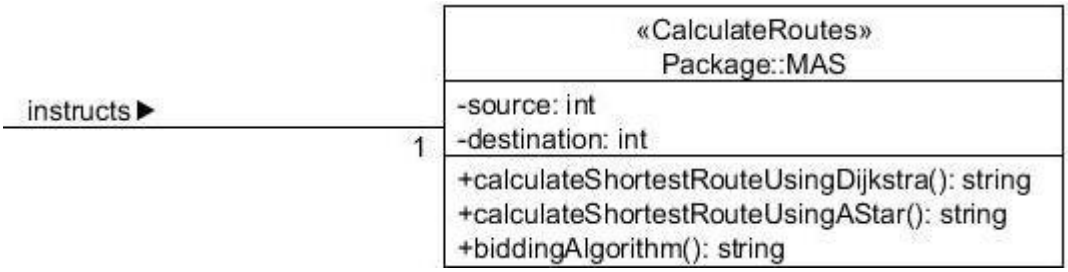
Figure 11 Message Class



Calculate Route class: this class implements three algorithms. First the Dijkstra algorithm to calculate the shortest route between a source segment and a target segment. Dijkstra algorithm is a special case of the A* graph search algorithm where the heuristic always returns zero. This class will also implement the A* pathfinding algorithm (Hart, Nilsson and Raphael, 1968) to achieve better performance and accuracy. Both of these algorithms, Dijkstra and A*, will be implemented in a distributed manner. In addition to these algorithms, this class will implement a bidding algorithm which given various multi-modal routes from a given source to a destination will negotiate a combined cheapest travel route (see section 3.3). The agent environment instructs this class to calculate travel routes.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

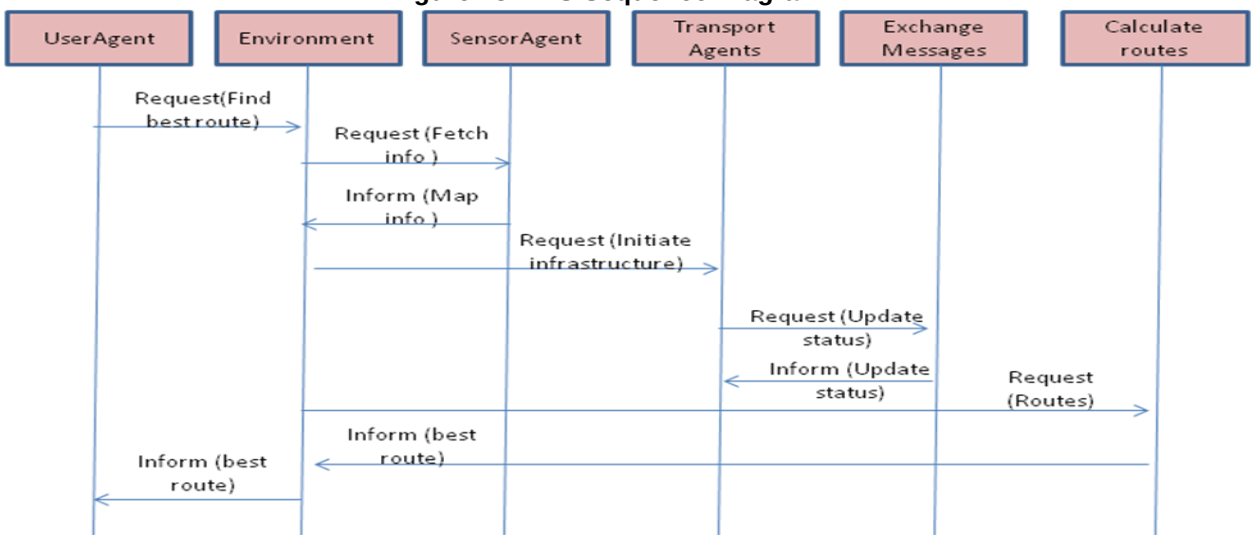
Figure 12 Calculate Route Class



3.2.2 Sequence Diagram

The sequence diagram describes the sequence of operations and actions of our multi-agent system, how they interact with each other, and in what order. The procedure starts with a user agent capturing a user request on behalf of an Android device end user (e.g. find me the most efficient travel route from a start point X to a destination point Y). This request is then transferred to the environment which sends a request to the sensor agent to fetch map information and current traffic updates from the traffic simulation model (WP3). The environment uses the latest map and traffic updates to initiate and populate all transport agents. The transport agents then communicate together in a round robin manner to update each other of their new cost. Once the message exchange process has concluded, the environment initiates a module to calculate the most CO_2 efficient route. This route is then communicated to the user agent which sends it to the designated end user. The operations and message sequence diagram is depicted in Figure 13.

Figure 13 MAS Sequence Diagram



3.3 Algorithms

The MODUM multi-agent system endeavours to reduce CO_2 emissions of urban transport vehicles, primarily cars by reducing their idling times and journey times, and minimising their use in favour of other sustainable forms of transport covering the same intended journeys. This will ultimately improve the overall commuting experience and commuter perception toward sustainable transport. To this end our multi-agent system will simulate the transport infrastructure, where each transport segment will have a composite cost assigned to it, use the available real-time traffic information, and propose the most CO_2-efficient multi-modal journey route. As such the cost in our context represents CO_2 emissions per segment. The

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

algorithms in our multi-agent system will find the path(s), from source segment to target segment, with the least total CO_2 emissions.

The cost of using a particular segment of the transport, be it road, bus, rail, cycle or pedestrian, is determined by a combination of factors such as:

- Type of vehicles flowing in the segment, including engine size and fuel type
- Length of segment
- Type of segment (e.g. uphill, downhill, flat)
- Traffic flow rate (i.e. number of cars per hour - characterising either free traffic flow or a congestion)
- Average vehicle speed
- Delays on the segment
- Time of travel (e.g. departure time)
- Weather condition (e.g. vehicle engines require more energy during winter).

The cost method will take into account as many factors as available from the traffic simulation model of WP3 to calculate the CO_2 emission levels for each segment. The emissions for a segment S, can be conceptualised using the abstract formula:

CO_2 Emissions (Segment S) = $\sum$ Weight (w) * Factor (f)

where the weight for each factor is an experimental decimal value ranging between 0 and 1, and will be configured during the simulation.

The multi-agent system will calculate for each transport segment a CO_2 cost and update this cost continuously as soon as the traffic conditions (e.g. average speed vehicle, flow rate, and delays) change. Messages are then exchanged between neighbouring transport agents to update each other with their new CO_2 cost. Now that the CO_2 cost for each segment is calculated, the multi-agent system can search for CO_2 efficient routes as per user travel requests and preferences.

The primary algorithm used to find the shortest route is the Dijkstra Algorithm (Dijkstra, 1959). The innovative aspect MODUM brings about would be to implement this algorithm in a distributed manner to provide route guidance to commuters. This graph search algorithm allows finding the shortest path from a given start node to a finish node within a graph containing nodes, edges and assigned weights. The concept of the algorithm is rather trivial yet very efficient; it starts from the source node, explores the neighbours and finds the path that has the cheapest cost between each node and its neighbouring nodes. Applying this algorithm to the context of MODUM is straightforward, where simply nodes represent transport segments and CO_2 emission represent assigned weights to using a particular segment. The algorithm then searches the graph to find the shortest route between any two segments.

The algorithm works in the subsequent logical manner:

1. Start the search from the initial (or source) segment, set its cost to zero, and make it the current segment

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

2. Mark all transport segments as unvisited except the initial segment, and add them to a set of unvisited segments
3. For the current segment, visit all of its unvisited neighbouring segments and calculate the accumulative cost (where new accumulative cost to neighbour = accumulative cost of current segment + cost of unvisited segment)
    a. If the new accumulative cost to neighbour is less than a previously-recorded accumulative cost to the neighbour then override that the accumulative cost
4. After visiting all neighbouring segments, mark current segment as visited and remove it from the set of unvisited segments
5. If the current segment is the target segment, then stop the algorithm. Trace back to print the shortest path.
6. Set the unvisited segment with the smallest cost from the unvisited set as the current segment, and repeat step 3

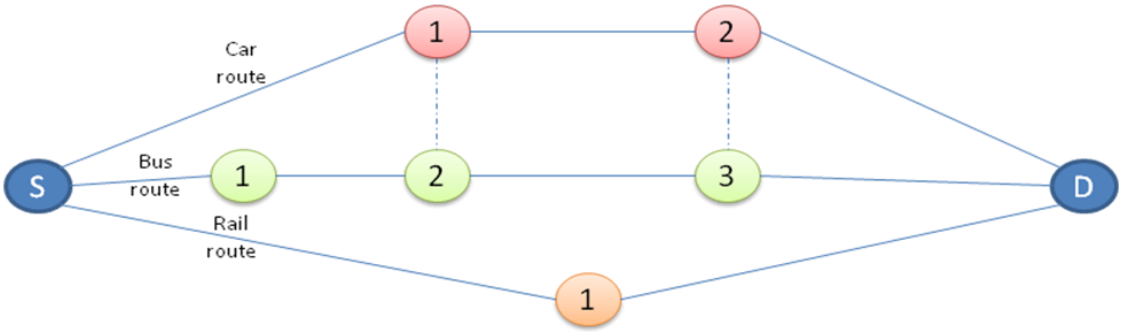
In later stages, we will also implement the A* algorithm (Hart, Nilsson and Raphael, 1968) which is well known for its accuracy and performance. Comparing the Dijkstra algorithm and A* algorithm in the context of MODUM, were real-time travel guidance is required, would yield interesting findings. Performance comparison between these two algorithms will be detailed in deliverable D2.2.

In addition to using CO_2 emissions as the cost for each segment, another innovative aspect in our multi-agent system lies within a bidding algorithm which, given the cost of concurrent multi-modal transport segments (e.g. road, rail and bus segments), will negotiate the optimal travel route for the commuter. In this respects, transport segments, thus agents, will compete by lowering their respective cost to be part of the travel route. To clarify the concept of the bidding algorithm we will give an example.

Given an urban transport network (Figure 14), there exist three main multi-modal routes between a source segment (s) and a destination segment (d): a car, bus and a rail route. The commuter can also combine, for instance, the use of car and bus as indicated by the dotted line connecting the car route and bus route. So in that sense the commuter can switch from using a car to a bus or vice versa at the connected transport segments as they belong to the same geographical location within the transport map. Herein we assume that the commuter has access to three modes of transport. Using any of the potential travel routes (car, bus, rail route or combined) comes at a cost (i.e. CO_2 emissions). The bidding algorithm comes in handy in situations where there exists more than one possible multi-modal route leading to a particular destination. It aims to leverage and minimise the cost of travelling from a given source to a given destination by studying the offers provided by different transport agents. In the diagram below, it studies the cost of going from S to D by all transport agents and chooses the cheapest bids. For example, the bus agents could bid to take the commuter from the source segment to bus segment 2 cheaper than what the car agents offer to take to car segment 1, where car segment 1 and bus segment 2 belong geographically to the same network segment. The bidding algorithm will then study the offer provided by car agents to take to car segment 2 against the offer provided by bus agents to take to bus segment 3, and choose the cheapest. The bidding algorithm continues the search and studies the offers until the cheapest combined multi-modal travel route emerges. For practicality reasons, the algorithm will take into account commuter preferences and restrictions, such as the inability to drive in which case it will not consider offers provided by car agents.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Figure 14 Bidding Example



3.4 Implementation and Initial Results

We are using Java version (JDK 1.7.0.04) to implement the agents of the system, the communication between the agents, and the calculation of the shortest travel route. For our first prototype we decided not to use any agent simulation framework or toolkit, such as JADE, for doing so would bring extra functionalities and features that would not be beneficial for our multi-agent system but would instead decrease the efficiency and performance of the traffic management simulation. Using solely Java as the implementation language brings about two main advantages:

- Enables tracking the behaviour of every single agent where the execution of agents is implemented in a round robin manner; thus enabling us to know what every agent is doing in the system at any particular point.
- Enhances performances and eliminates the need to burden the multi-agent system with unnecessary functionalities.

For the initial MAS prototype, the following agent classes have been created and implemented:

- Interface agent class describing behaviour of transport agents
- Transport agent classes describing the types and behaviour of agents
- Communication classes to manage message exchanges between agents
- Other classes, e.g. for calculating the shortest travel route using the Dijkstra algorithm

As an initial step the first version of the multi-agent system was tested using two scenarios: a small scenario comprising five road segments and a realistic scenario comprising 28 road segments.

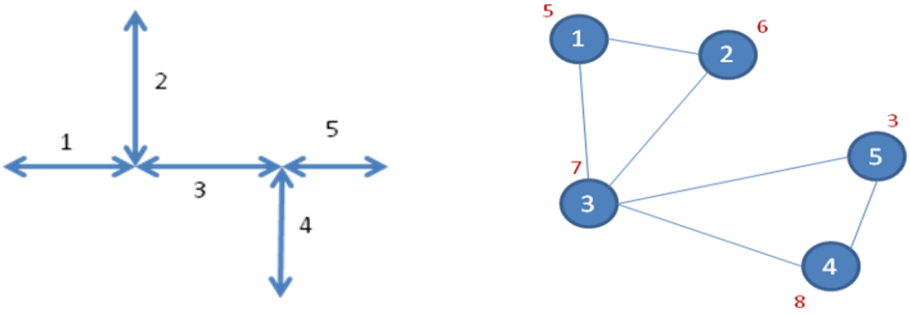
Example One

This small example includes 5 road segments where each segment is represented via a unique road agent with an ID (i.e. integer). The result is a small multi-agent system consisting of 6 road agents exchanging information about their cost, where cost is simply a function of accumulated length. As described earlier, each road agent holds information about itself (e.g. segment ID, length, number of lanes) and about the neighbouring agents (e.g. type of neighbours, their IDs and cost). The left diagram (of Figure 15) shows a potential road structure, whilst the right diagram (of Figure 15) shows an inferred road graph constructed by the MAS to search for the shortest travel route. The nodes in the road graph

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

represent the road segments with the assigned values representing the length of each segment. These values also represent the cost of using a particular road segment.

Figure 15 A Small Test Scenario

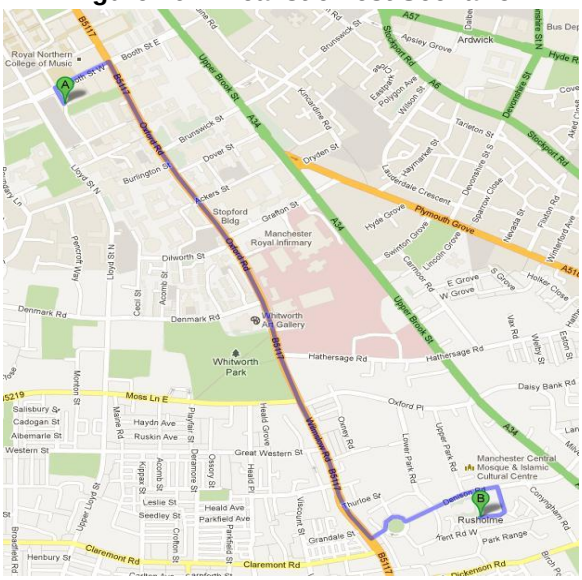


Example Two

This more realistic scenario represents a road area from the Manchester (UK) map and consists of 28 interconnected road segments. Each road segment is assumed to contain two lanes for driving in both directions. Each road agent represents only one segment and has at least three neighbouring road agents connected to it. At the moment, the cost of using one particular travel route is a function of accumulated lengths of the connected road segments leading to the destination segment. In the next prototype versions we will enhance the cost to include other properties such as CO_2 emissions, delays and traffic flow. Any one road agent has the following attributes:

- ID
- Name of segment (e.g. road name)
- Role (e.g. road agent, rail agent, etc)
- Neighbours
- Length
- Cost
- Number of Lanes
- Outbox Messages
- Inbox Messages

Figure 16 A Realistic Test Scenario



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

In both scenarios, the multi-agent system was able to successfully simulate the road infrastracture and calculate the shortest travel route between any two segments (a start segment, a finish segment) using the Dijkstra algorithm. The output is a set of strings representing the IDs of the cheapest travel route from a given source segment to a destination segment.

3.5 Future Implementation Plan

For the next versions of our MAS prototype we detail the implementation steps, features to be added to the multi-agent system and appropriate delivery dates.

Table 2 Future Implementation Actions for MODUM Multi-Agent System

Implementation Action	Date
-Populate MAS with WP3 simulation map data	15/11/12
-Random generation of user requests (e.g. sources and destinations)	20/11/12
-Dynamic traffic updates (update requests and responses) from traffic model simulation (WP3) to MAS	30/11/12
-Enhancement to the cost function by including historical CO_2 emissions, traffic flows and delays for transport segments	30/11/12
-Implement rest of infrastructure - bus maps, rail maps. Each agent will maintain one timetable for the buses / trams that go through it. This will be stored in the belief set of the agent. A typical time table will contain information about service arrival times and providers' names.	20/12/12
-Implement rest of infrastructure- bicycle maps, pedestrian maps	20/01/13
-Reasoning engine and bidding algorithm	10/02/13
-Reading actual user requests and preferences from a web server	20/02/13
-Generation of multi-modal route guidance	15/03/13
-Optimisation of multi-modal route guidance taking into account user preferences and constraints	30/03/13

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

4 MODUM Ant-based Traffic Management Model

The ant-based traffic management model provides a traffic coordination infrastructure. The infrastructure supports multi-modal traffic and goes beyond an ICT infrastructure, offering communication and computation services, and traffic coordination related services. Simultaneously, it does not impose specific choice mechanisms on the user. The model provides an infrastructure on which those choices can be executed. This system visualizes (i.e. makes observable by humans and software processes):

- Current and past traffic situation (track and trace).
- Predicted traffic situation accounting for user intentions. For instance, accounting for user intentions allows visualizing to what extent the traffic participants have managed to coordinate cooperatively before intervening.
- On-line searchable solutions space. This allows users to find and evaluate alternatives, accounting for the predictions.

The ant-based traffic management model generates traffic predictions given:

- A high penetration/participation (think of mobile phone)
- Travellers self-prescribe their routing and their timing

In contrast to many state-of-the-art models, this model aims to coordinate the traffic flow in real-time, i.e. an online model. Users are not known upfront and appear on the traffic management system as they plan a trip. Real-time traffic conditions are taken into account, e.g. car accident, weather conditions, etc.

The solution is centred on a high participation mode and subsequently expands the applicability range of its systems and mechanisms later (e.g. for lower penetration/participation).

4.1 Architecture

The developed architecture focuses on the distributed nature of traffic systems and the real-time operation mode.
A traffic system comprises a set of autonomous entities. All entities have to some extend a degree of freedom;

1. Car drivers have the freedom to select their personal route and timing
2. Traffic infrastructure cannot be centralised entirely.

Still it is opportune to coordinate these "selfish" users. This is mainly done by providing information and incentives.

Two main architectural assets enable the distributed real-time coordination, the PROSA++ architecture and the delegate MAS pattern. Finally the application architecture refers to the structure of the application itself.

4.1.1 PROSA++

The ant-based traffic management application uses a holonic architecture (Koestler, 1989; Babiceanu, 2006; McFarlane, 2000). A holonic architecture is a multi-leveled hierarchy of semi-autonomous subwholes, branching into sub-wholes of a lower order. Sub-wholes on any level of the hierarchy are referred to as holons (Koestler, 1989). In the context of traffic, a

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

holon is defined as an autonomous and cooperative building block of a traffic control system for informing and guiding traffic entities.

The holonic architecture used in this management model is adopted from a well-studied and widely used architecture in manufacturing control systems, PROSA (Van Brussel, 1998). The PROSA architecture was originally developed in the manufacturing domain. It has also been applied in other application domains (railway systems (De Swert, 2006), logistic systems (Van belle, 2011), robotic systems (Philips, 2011), and others (Van Belle, 2012)). This architecture is further elaborated towards PROSA++ which appeared to be well suited for traffic control systems.

The PROSA++ reference architecture

The PROSA++ architecture identifies four types of holons as indicated in Figure 17. We can distinguish products and resources. Products relate to any activity in the world of interest, in this case traffic users. Resources relate to the enabling entities in the world of interest, in this case the traffic infrastructure, e.g., links, nodes, railway company, etc
Note that due to the origin of PROSA++ (manufacturing), the terminology product and resource may seem confusing. Nevertheless, the concepts remain valid and have been proven outside the manufacturing domain (Van Belle, 2012).

The ResourceInstance holon reflects a specific part of the traffic infrastructure. This holon contains information about the physical entity, the present state, future states and what-if functionality.
We consider two classes:

- Route infrastructure; nodes and links
- Multi-modal transport e.g. tram, bus.

The class, route or multi-modal transport is encapsulated into the ResourceInstance holon. The same interface is used to the remaining system. Preferences to use one or another ResourceInstance holon are expressed in terms of capability (e.g. amount of luggage), availability and trust (reputation from one user towards a ResourceInstance holon).

A ResourceType holon holds policies regarding a specific resource entity or a group of resources. Examples of these policies are: max/min speed on a link, bus lanes,… The policies are communicated to the ResourceInstance in order to ensure a correct and desired behaviour on the traffic network entity.

The ProductInstance holon corresponds to a request for a trip in the traffic network, originating from a traffic user. This holon is responsible for planning and guiding the assigned trip correctly and on time. The ProductInstance holon searches and evaluates candidate routes and potential multi-modal alternatives. A selection of the candidates is presented to the user while the final selection – the intention – is the responsibility of the user. The ProductInstance holon reflects this decision to the other holons.

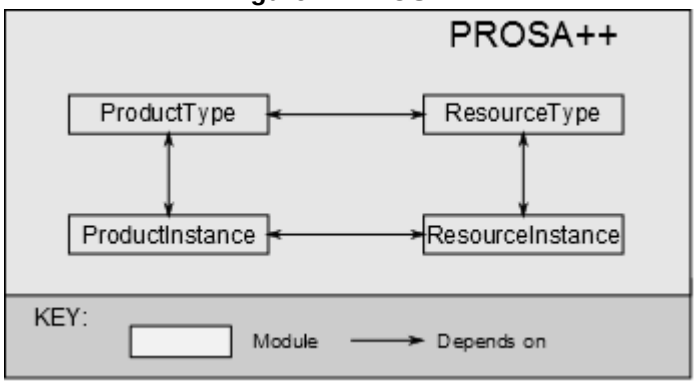
ProductType holons hold all policies regarding a traffic user or a group of users. These policies can be various, some examples:

- Preferences regarding multi-modal transport, availability of car, …
- Preferences regarding distance
- Preferences regarding route types (e.g. scenic route), stops allowed / preferred

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

These policies are exchanged with the ProductInstance holon to guide the search for good journeys.

Figure 17 PROSA++



Motivation

Using a holonic architecture requires an identification of the different types of holons. Responsibilities need to be assigned to each holon type. Also relationships between the holons need to be defined clearly. PROSA++ is a reference architecture for holonic architectures that reduces the impact of changes in decision making by separating concerns. The reference architecture allows to (Verstraete, 2008; Van Brussel, 1998):

- Separate the "resource" (traffic infrastructure) aspects from the "product" (trip planning and driving to its destination) specific aspects. Typical for traffic control is the difference in goal between the "selfish" traffic user and the traffic infrastructure as system-wide optimiser.
- Separate the necessary modules, which are generic, from the optional modules, which can be domain specific. ProductType, ProductInstance, ResourceType and ResourceInstance all hide the specific technical details from each other.
- Separate the structural aspects of the architecture from the algorithmic aspects. Existing scheduling and planning algorithms can be integrated without affecting the basic architecture.

4.1.2 Delegate MAS

Delegate MAS is an architectural pattern that allows an agent to delegate a responsibility to a swarm of lightweight agents to support this agent in fulfilling its functions (Holvoet, 2007; Verstraete, 2008). The issuing agent can delegate multiple responsibilities, each of them applying the delegate MAS pattern. The agent may use a combination of delegate multi-agent systems to handle a single responsibility. The delegate MAS may also provide services to other agents.

The "Delegate MAS" pattern translates insights from the food foraging behaviour in ant colonies into the software design (Valckenaers, 2005).

- Refresh-and-evaporate: ants deposit pheromone trails that evaporate unless refreshed by ants walking along such a trail. This translates into: "All information in the traffic management system that is subject to real-world dynamics has a finite lifespan." For instance, trip reservations need to be reconfirmed regularly or they are discarded by the resource (e.g. parking space) concerned.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

- The environment contributes to the solution: ants deposit their pheromone trails on the real-world environment, allowing them to cope with almost any geometrical complexity by means of a single simple procedure. Translating this mechanism, an Environment is created that mirrors the world-of-interest in software (physical traffic infrastructure), and the environment components will be made intelligent/cognitive as needed or opportune.
- Swarming: from simple ant behaviours emerges a sophisticated well-performing behaviour for the colony overall. However, the colony needs numerous cheap ants to achieve this. In the model, traffic users create swarms of lightweight agents – called ant agents – that travel virtually across the environment. Because they are virtual entities, ant agents are cheap and can be numerous. These swarms, performing services on behalf of agents, are called a delegate MAS (Holvoet, 2007).
- Computational efficiency: our ant-like design has a low-polynomial computational complexity in function of the effort needed for the primitive actions of virtual travel through the environment.

Delegate MAS architectural pattern

An architectural pattern is a description of element and relation types together with a set of constraints on how they may be used.
The delegate MAS pattern consists out of three elements: the agent, the ant and the environment (Figure 18).

Environment: the environment is a software representation of the world of interest. The environment contains a directed graph, which may change over time. The nodes in the graph represent a location in the world of interest and the edges represent the connections between different locations. An agent is located on a node in this graph.
The environment contains a pheromone infrastructure. Ants deposit, observe and modify pheromones in the pheromone infrastructure.

Agent: an agent delegates a responsibility to a swarm of lightweight agents. This delegation involves two main responsibilities: maintaining the swarm of ants and interpreting the results made available by the ants in the environment.

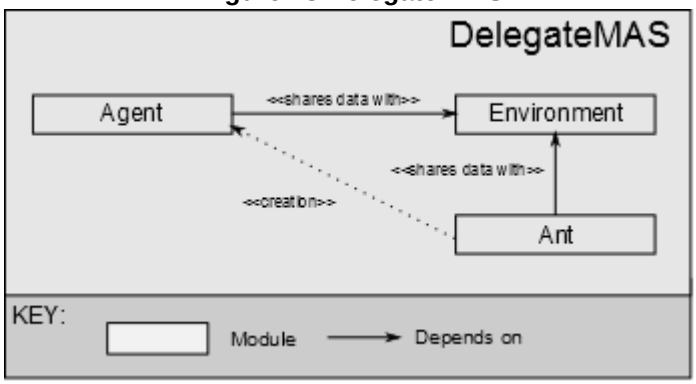
- The agent is responsible for maintaining the swarm of ants. This responsibility involves maintaining the population size and the diversity of the swarm. An individual ant is not aware of these swarm properties.
- Interpreting the results involves fetching the pheromones on the node on which the agent is located and interpreting these pheromones. The interpretation depends on the responsibility that is delegated by the agent. Note that delegation is not the same as total dependency. The swarm of ants supports the agent by providing a service. If this service is not delivered properly, the agent should do a best effort to cope on its own.

Ant: an ant is responsible for executing a task that serves a responsibility of the issuing agent. For instance, the ant explores one possible solution; the ant follows one possible path to find food, etc.

The ant is created and initialised from the agent and travels autonomously. Each ant has its own lifecycle and may only perform a bounded computational effort within its bounded lifetime and has a bounded footprint (memory).

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Figure 18 Delegate MAS



Motivation

Traffic is characterised by its distributed and dynamic nature. An agent needs to be capable of making decisions that are adapted to the events happening in the environment, i.e. the traffic network. The delegate MAS pattern allows an agent to exploit the detailed and up-to-date information in this environment to adapt his intentions to these events.

4.1.3 Application architecture

The application architecture shows the global structure of the application. It consists out of three layers, an agent layer, an environment layer and dispatching layer. The application architecture also indicates the relationship between the PROSA++ architecture and the Delegate MAS pattern.

Holon = Agent + IBeing

In summary, a holon is composed out of two software components: an agent and an intelligent being (IBeing). The intelligent being only reflects the corresponding entity while an agent has the responsibility to take decisions on behalf of the corresponding entity. A typical example of an intelligent being in the traffic domain are the models representing the flow behaviour of a link depending on the arriving flow.

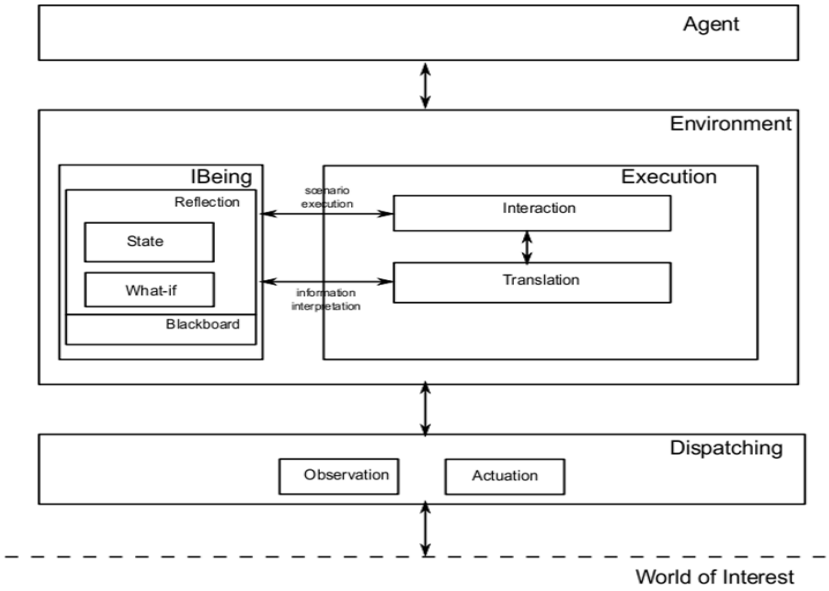
A holon resides both in the "Agents" module as well in the "Environment" module, i.e. the agent component of a holon resides in the "agents" module while the IBeing resides in the "environment" The information flow between the Agents and the IBeings is channelled through an execution module, ensuring the correct execution of a scenario (ants updating their state) and information interpretation.

Dispatching

The dispatching module is the communication channel between the world of interest and the ant-based application.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Figure 19 Application Architecture



4.2 Software entity models

The ant-based traffic management model uses local models to make a short-term forecast of the traffic situation. These models represent one particular entity in the traffic system. As mentioned before: two entity types are considered:

- Resources: resource are entities which are part of the traffic network, e.g. links, nodes, public transport organisations, etc
- Products/activities: activities are entities making use of the traffic network, e.g. a person driving to his/her destination

The representation of congestion and congestion dynamics is essential to ensure the generation of a trustworthy short-term forecast. First-order traffic flow theory (as described in section 2.2) is universally acknowledged to represent traffic propagation and congestion dynamics. Figure 20 shows the models adopted from the 1st order order traffic flow theory The sub-models are used independently in order to model local travel behaviour on nodes and links.

The selection of the first-order traffic flow theory is a balance between on the one hand simple - less realistic - approaches such as travel time functions, vertical or horizontal queuing (Corthout, 2012; Nie and Zhang, 2005; Mun, 2007) and on the other hand second-order traffic flow theory.

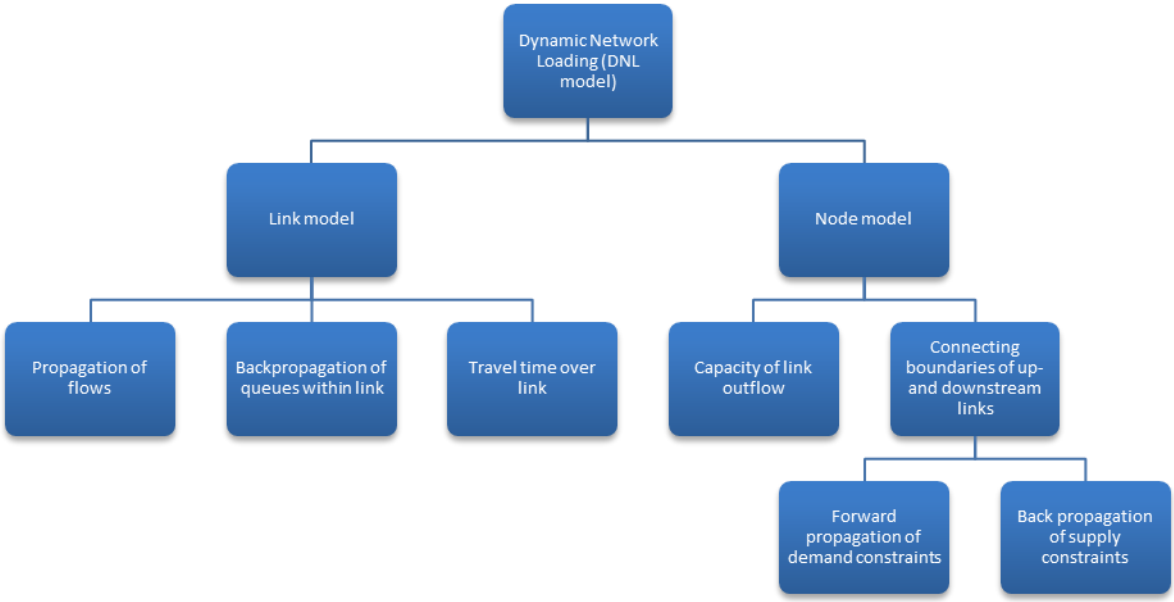
Second-order traffic flow theory - first introduced by (Payne, 1971) - differs from first-order theory in that the fundamental relationship is considered to be not a stationary, but only an equilibrium relation. Additional traffic phenomena such as acceleration and deceleration and traffic instabilities (e.g. in the form of stop-and-go waves) can be modelled, which are not included in first-order models.

Second-order traffic theory is not relevant for two reasons:

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

- In this particular application, a real-time application, the equilibrium is not static. As trip requests appear and traffic conditions change, the equilibrium is invalidated and a new equilibrium should be calculated. Consequently, the error due to the uncertainties tends to be higher than the gain in model precision.
- The ant-based approach requires computational efficient local models, to enable the desired properties of an ant-based application.

Figure 20 DNL Model



4.2.1 Fixed-point Formulation

From a computational perspective, some first-order traffic flow models are forced to operate at very small time steps. These small time steps increase the overall computation time tremendously. However, this criticism does not carry over to numerical schemes following the variational formulation of kinematic waves (Daganzo, 2005), or the fixed-point formulation (Gentile et al., 2007), both of which avoid this constraint.

In numerical analysis, fixed-point iteration is a method of computing fixed points ($F(x) = x$) of iterated functions. More specifically, given a function defined on the real numbers with real values and given a point in the domain, the fixed point iteration is $F(x_i) = x_{i+1}$ where $i = 0, 1, 2,\ldots$ which gives rise to the sequence $x_0, x_1, x_2,\ldots$ The sequence is expected to converge.

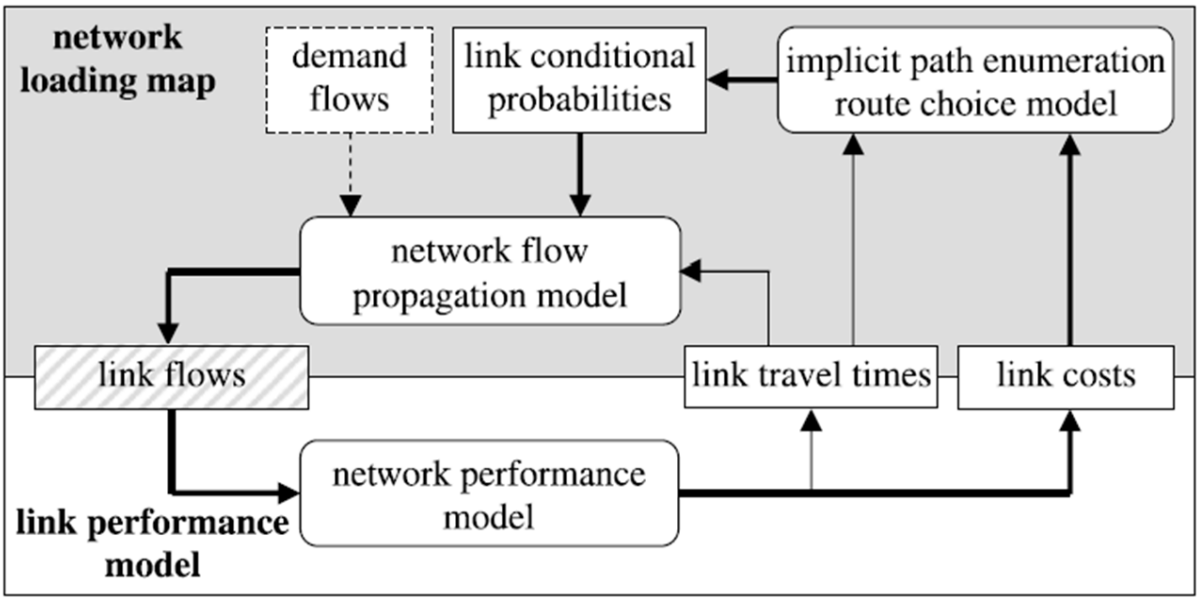
The fixed-point formulation of Gentile et al. (2007), as illustrated in Figure 21, operates on a function redistributing the link flows. This function both

- Propagates flow through the network according to the previous calculated link travel times.
- Checks flow constraints over the network and redistributes traffic flow accordingly. New travel times are calculated if the traffic flow is redistributed.

This iterative procedure is executed until equilibrium is reached, i.e. traffic flow does not change anymore.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Figure 21 Fixed Point Formulation



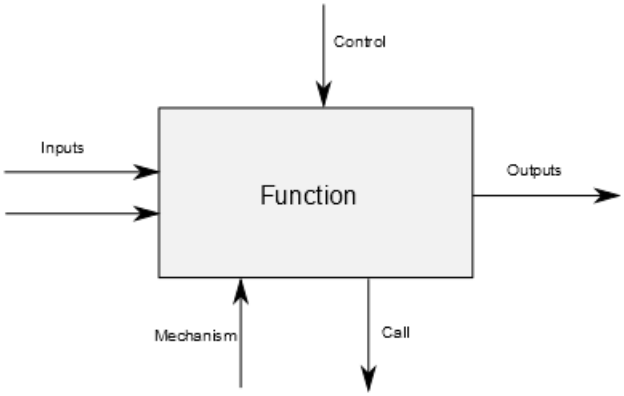
4.3 Process views

This section gives a detailed overview of the different processes and their relations. We have chosen to use the IDEF0 notation (Ross, 1985). IDEF0 was derived from a well-established graphical language, the Structured Analysis and Design Technique (Ross, 1985). IDEF0, integrated definition for function modelling, is a method designed to model the decisions, actions, and activities of an organization or system.

IDEF0

Figure 22 shows some basic building blocks of IDEF0[1]. The "function" is an activity, process, or transformation (modelled by an IDEF0 box). The function is described by their inputs, outputs, controls, and mechanisms (ICOMs). The place of the arrows has a dedicated meaning, as indicated in Figure 22.

Figure 22 IDEF0 notation



4.3.1 Application process

The application process gives an overview of the high level processes active in the application and their relations.

[1] For further reading please refer to http://www.idef.com/IDEF0.htm

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Creation

Both OrderInstances and ResourceInstances are created from a proxy who provides the link between the ant-based application and the world of interest (WoI). The necessary data is provided (e.g. information about link properties) but is also update regularly.

The advice from the Order- and ResourceType is modelled as a mechanism for respectively the OrderInstance and the ResourceInstance.

Ants are created at regular times. The ant-based traffic management model makes use of three ants:

1. Explorer ants: each explorer ant is representing one specific trip. The explorer has the responsibility to search for desirable journeys, both on the traffic network as well as multi-modal alternatives.
2. Intention ants: The intention ants do propagate a declared intention. It can be compared to the "Network flow propagation model" of the fixed-point formulation (Figure 23).
3. Flow intention ants: Flow intention ants represent an aggregate of several vehicles. It travels up-or downstream and ensures all constraints on adjoining links or nodes. This corresponds to the "network performance model" in the fixed-point formulation (Figure 23).

Note that the OrderInstance initiates (calls) ExplorerAnts and IntentionAnts while the ResourceInstance initiates FlowIntentionAnts.

Feedback

There are several feedback loops in this schema:

- The ExploreAnt notifies the OrderInstance about a feasible journey
- The OrderInstance provides the user with a selection of candidate journeys. A selection is made by the OrderInstance
- The FlowIntentionAnt provides feedback to the ResourceInstances about their out- and incoming flow

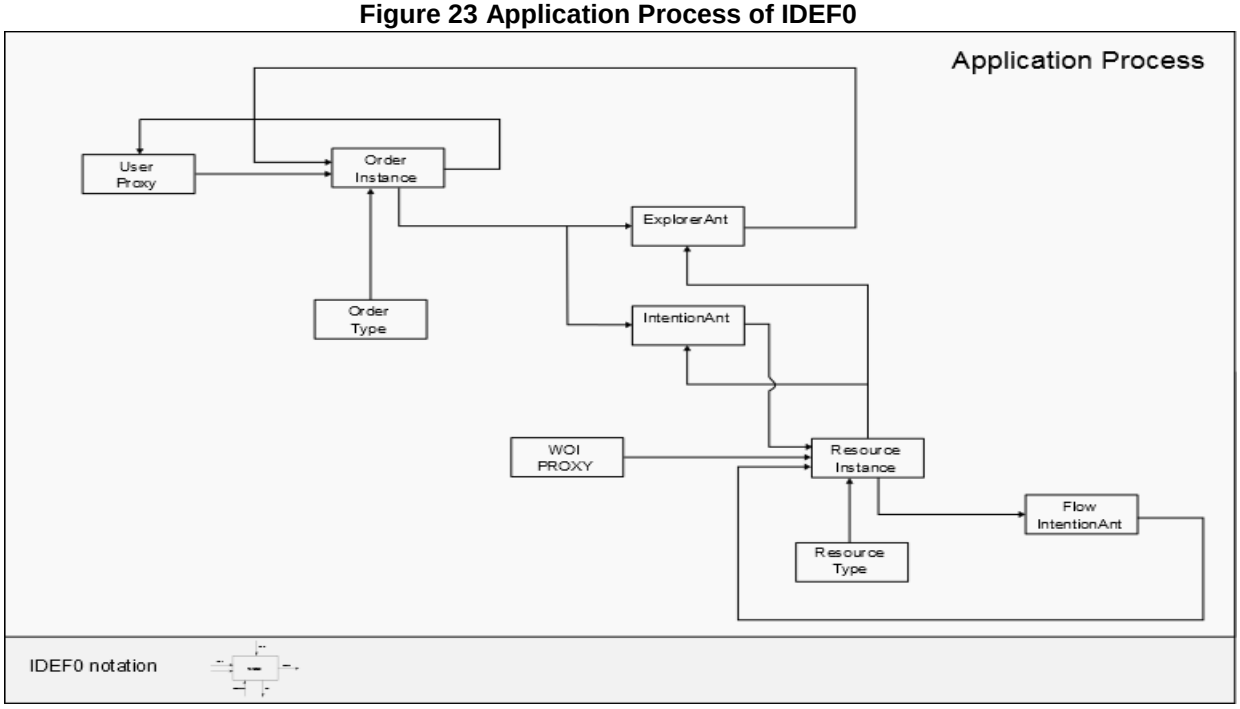
Equilibrium

Given the dynamic situation it is not feasible to search a static equilibrium. Nevertheless ResourceInstances should tend to a new equilibrium each time the situation is changed. Therefore each ResourceInstance individually keeps track of a (in)stability measure for their incoming and outgoing cumulative flow.

Depending on this measure, the ResourceInstance generates at higher or lower frequency FlowIntentionAnts. Also, a FlowIntentionAnt is aware of this measure and determines whether to stop propagating or not.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

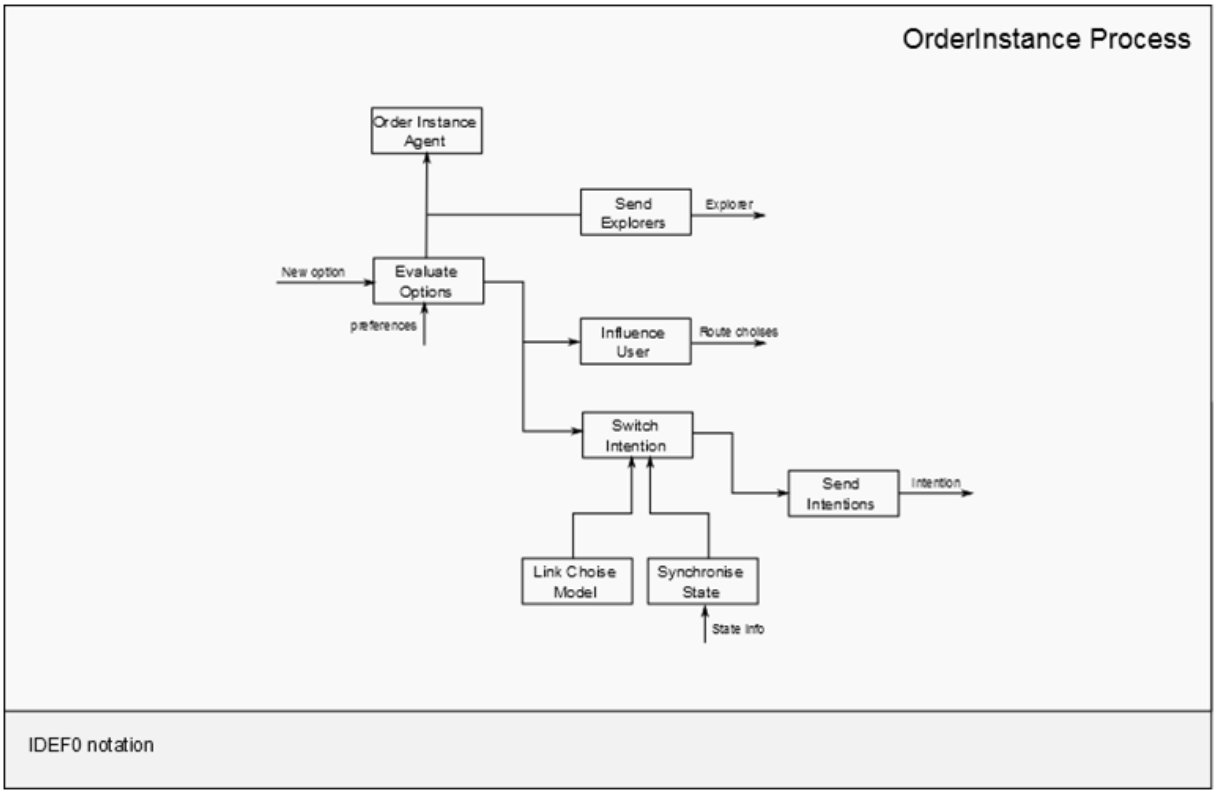
Figure 23 Application Process of IDEF0



4.3.2 Order Instance/Type process

This view shows a detailed view of the OrderInstance process. The OrderInstance is mainly a manager, initiating explorers, intentions and providing users with the necessary data.

Figure 24 Order Instance Process



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

4.3.3 Resource Instance/Type process

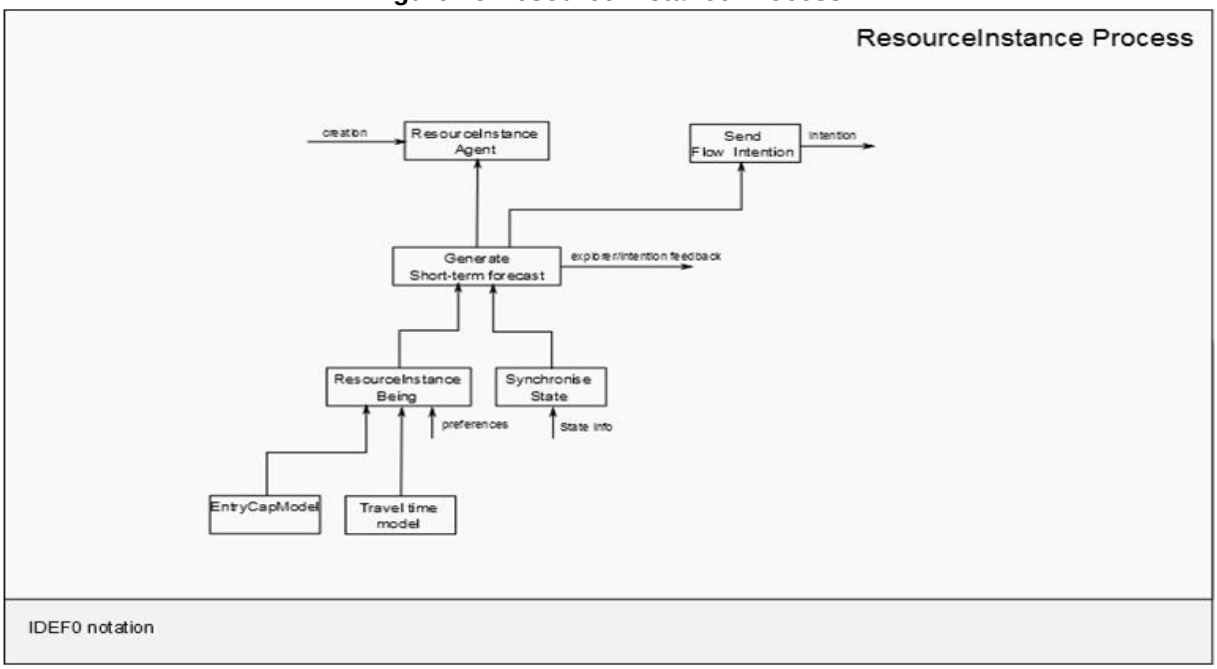
This view shows a detailed view of the OrderInstance process.

Short-term forecast

The short-term forecast, i.e. the expected flow, is generated by means of the current state and 1st order traffic flow models.

- Using the current state as start point gives an automatic recalibration when the reality deviates from the last know short-term forecast.
- The intelligent being of the resource instance models the local traffic flow behaviour according to the relevant 1st order traffic flow models.

Figure 25 Resource Instance Process



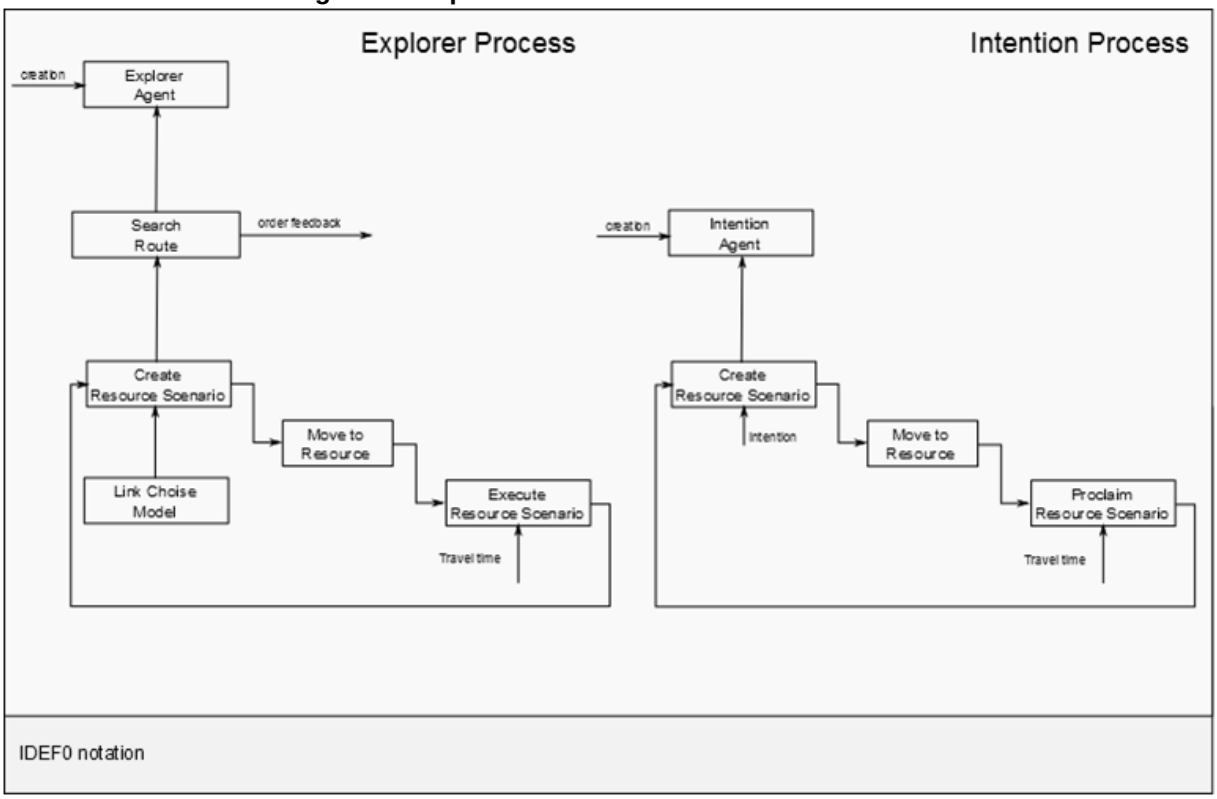
4.3.4 Explorer process / Intention process

Figure 26 shows the explorer and intention process in detail. There are two main differences between the explorer and intention process:

- The explorer process makes use of the link choice model to determine the next link/node while the intention process follows the last known intended journey.
- While the explorer process only executes a scenario (does not change the short-term forecast), the intention process proclaims the intention.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

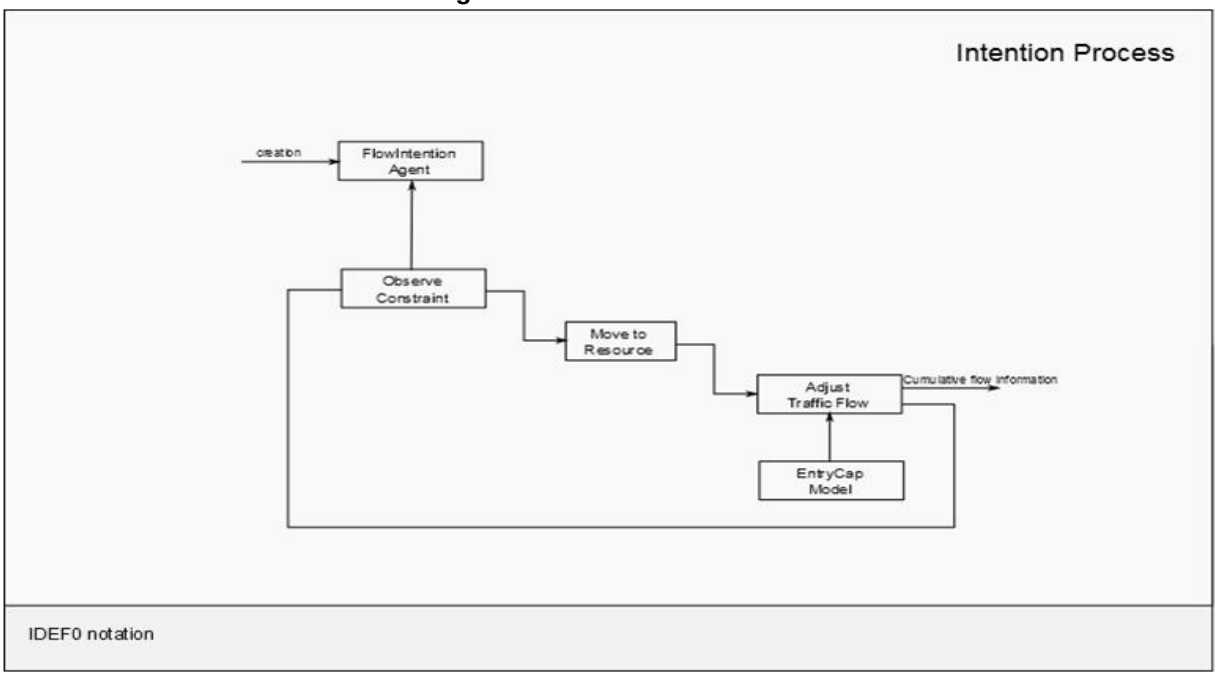
Figure 26 Explorer Process and Intention Process



4.3.5 Flow Intention process

The flow intention process has a similar structure as the other ants but do represent multiple vehicles. As mentioned above, the flow intention process adjusts the cumulative flows in order to comply with all constraints.

Figure 27 Intention Process



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

4.4 Algorithms and Implementation

4.4.1 Erlang

In the search for a language and multi-agent framework, erlang (ERicson LANGuage; www.erlang.org) is selected as being a language with the necessary multi-agent framework functionalities. "Erlang/OTP"–developed by Ericsson AB and known to be very robust– is industry-hardened. More specifically, we have been experiencing the following interesting properties.

Actor model

Erlang provides massively parallel and distributed processing. Every process of the Erlang virtual machine is an actor.

- Each actor executes independently and communicates only with one-way messages
- Actors can create other actors
- When an actor finishes its computation, it disappears

Erlang has no shared memory between processes. Data exchange is straightforward as the locality of the data is ensured.

Note that Erlang processes run inside one or more virtual machines on one or more nodes of a network, so they do not map to OS processes.

Integration

Erlang provides easy integration with other systems through services (including legacy systems). Its baseline, OTP – thanks to its industrial roots and especially its telecoms origin – offers a rich set of libraries and middleware to connect with external systems.

Run-time configuration

Erlang provides the mechanisms to make software systems reconfigurable at run-time. Replacement of software processes and modules while the systems remains on-line and functioning is supported.

Pattern matching

Pattern matching enables the software modules and processes to interact flexibly when they recognise the structure of the information that is exchanged and trigger the associated actions.

Other properties

- Pragmatic; Academia - researching programming languages - typically consider it to be (too) pragmatic
- Lean and mean (small, can be learned in little time)
- High service availability (99.999% upwards)
- Distributed applications (high-performance computing, Cloud-based)
- Stability of the virtual machine (BEAM is better than Java VM)
- Soft real-time
- Linkable to C-code and Java Code
- Open source

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

4.5 Future Implementation Plan

For the ant-based prototype development we detail the implementation steps and appropriate delivery dates.

Implementation Action	Date
PROSA++ and Delegate MAS Infrastructure	15/12/12
Random generation of user requests (e.g. sources and destinations)	15/01/13
Populate MAS with WP3 simulation map data	15/01/13
Dynamic traffic updates (update requests and responses) from traffic model simulation (WP3) to MAS	15/01/13
Implement multi-modal resource holons • Tram / bus alternatives • bicycle maps, pedestrian maps	30/01/13
Reading actual user requests and preferences from a web server	28/02/13
Optimisation of multi-modal route guidance taking into account user preferences and constraints	30/03/13

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

5 Conclusion

Deliverable D2.1 describes the architectural outline and inner workings of two distinct traffic control and management simulations both competing to tackle the CO_2 emission problem within MODUM: an agent-based system and an ant-based system. The first system models transport segments and aims to optimise their use whilst the second system models vehicles and aims to optimise their driving behaviour. Each model offers a unique distrusted solution to search for the most CO_2-efficient travel routes between a travel source and a travel destination within a transport network.

Despite their differences, the two traffic management models can be complementary indeed within the context of MODUM where the multi-agent system can be used to calculate a best travel route which is then used to guide the search of the ant-based system. Essentially ants will use the solution of the multi-agent system as a starting point to explore the traffic network and optimise the solution further if possible. The synergy between the two models will be explained in detail in deliverable D5.1 of WP5.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

6 References

Babiceanu, R.F., and Chen, F.F., 2006. Development and applications of holonic manufacturing systems: a survey. Journal of Intelligent Manufacturing, 17(1), pp.111-31.

Bando, M., Hasebe, K., Nakayama, A., Shibata, A. and Sugiyama, Y., 1994. Jpn. J. Ind. Appl. Math.

Bazzan, A. L. C., Wahle, J., and Klugl, F., 1999. Agents in traffic modelling-From reactive to social behaviour. In Proc. KI: Adv. Artif. Intell., vol. 1701, pp. 303–306.

Bellman, R., 1957. A Markovian Decision Process. Journal of Mathematics and Mechanics 6.

Bellomo, N., Marasco, A., Romano, A., 2002. From the modelling of drivers' behaviour to hydrodynamic models and problems of traffic flow. Nonlinear Anal. RWA, 3, pp. 330–363.

Brackstone, M., and McDonald, M., 1999. Car-following: a historical review Transportation Research Part F, 2, pp. 181–196.

Burmeister, B., 1996. Models and methodology for agent-oriented analysis and design. In: K. Fischer, editor. Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems. DFKI Document D-96-06.

Burmeister, B., Haddadi, A., and Matylis, G., 1997. Application of multi-agent systems in traffic and transportation. IEEE Proceedings on Software Engineering, 144 (1), pp. 51–60.

Chen, B., Cheng, H. H., and Palen, J., 2009. Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. Transp. Res. Part C: Emerging Technol., vol. 17, no. 1, pp. 1–10, 2009.

Chen, B., and Cheng, H. H., 2010. A review of the applications of agent technology in traffic and transportation systems. Trans. Intell. Transport. Sys. 11, 2, 485-497.

Claes, R., and Holvoet, T., 2012. Cooperative ant colony optimization in traffic route calculations. Practical Applications of Agents and Multiagent Systems.

Corthout, R., 2012. Intersection Modelling and Marginal Simulation in Macroscopic Dynamic Network Loading.KU Leuven.

Daganzo, C.F., 2005. A variational formulation of kinematic waves: Solution methods. Transportation Research Part B: Methodological, 39(10), pp.934-50.

Dam, K., and Winikoff, M., 2004. Comparing agent-oriented methodologies. In P. Giorgini, B. Henderson-Sellers, and M. Winikoff, editors. Agent-Oriented Information Systems, volume 3030 of Lecture Notes in Computer Science, pages. 78-93. Springer Berlin/Heidelberg.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Darbha, S., Rajagopal, K.R., Tyagi, V., 2008. A review of mathematical models for the flow of traffic and some recent results, Nonlinear Analysis: Theory, Methods and Applications, Volume 69, Issue 3, p. 950-970.

De Swert, K. et al., 2006. Coordination and control for railroad networks inspired by manufacturing control., 2006. IEEE Computer Society.

Di Caro, G., Dorigo, M., 1998. Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9(1):317–365.

Dia, H., 2002. An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. Transp. Res. Part C: Emerging Technol., vol. 10, no. 5/6, pp. 331–349.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1: 269–271.

FIPA Communicative Act Library Specification: Foundation for Intelligent Physical Agents, December 3, 2002.

Garcia-Serrano, A. M., Vioque D. T., Carbone, F., and Mendez, V. D., 2003. FIPA-compliant MAS development for road traffic management with a knowledge-based approach: The TRACK-R agents. In Proc. Challenges Open Agent Syst. Workshop, Melbourne, Australia.

Gazis, D., Herman, R., Rothery, R., 1961. Nonlinear follow-the-leader models of traffic flow. Operation Research, 9, pp. 545–567.

Gentile, G., Meschini, L., and Papola, N., 2007. Spillback congestion in dynamic traffic assignment: A macroscopic flow model with time-varying bottlenecks. Transportation Research Part B: Methodological, 41(10), pp.1114-38.

Gipps, P. G., 1981. A behavioural car-following model for computer simulation. Transportation Research Part B, 15, pp. 105–111.

Greenshields, B. D., 1935. A Study of Traffic Capacity. Highway Research Board Proceedings, Vol. 14, pp. 448–477.

Hart, P. E., Nilsson, N. J., Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107.

Holvoet, T., and Valckenaers, P., 2006. Beliefs, desires and intentions through the environment. Autonomous Agents & Multiagent Systems: Agent Theories, Architectures, and Languages, pp. 1052-1054.

Holvoet, T., and Valckenaers, P., 2007. Exploiting the environment for coordinating agent intentions. Springer-Verlag.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Hoogendoorn, S. P., and Bovy, P. H. L., 2001. State-of-the-art of vehicular traffic modelling. J. Syst. Control Eng., vol. 215, pp.283 - 303.

Intelligent Software Agents., 2010. Carnegie Mellon University, robotics institute. <http://www.cs.cmu.edu/~softagents/multi.html>.

Iglesias, C. A., Garijo, M., and Gonzalez, C. J., 1998. A Survey of Agent-Oriented Methodologies. In Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages (ATAL '98). Springer-Verlag. London, UK, 317-330.

Jennings, N.R., and Wooldridge, M., 1998. Applications of intelligent agents. Agent technology: Foundations, applications and markets. Springer-Verlag, New York.

Kerner, B. S., 2009. Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory. 1st ed. Springer.

Kinny, D., Georgeff, M., and Rao, A., 1996. A methodology and modelling technique for systems of BDI agents. In W. van der Velde and J. Perram, editors. Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World MAAMAW'96, (LNAI Volume 1038). Springer-Verlag: Heidelberg, Germany.

Koestler, A., 1989. The Ghost in the Machine. 2nd ed. London: Arkana Books.

Kukla, R., Kerridge, J., Willis, A., and Hine J., 2001. PEDFLOW: Development of an autonomous agent model of pedestrian flow. Transp. Res. Rec., vol. 1774, pp. 11–17.

Laird, J. E., Newell, A., and Rosenbloom, P. S., 1987. SOAR: an architecture for general intelligence. Artif. Intell. 33, 1 (September 1987), 1-64.

Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Lieu, H., 1999. Traffic-Flow Theory. Public Roads (US Dept of Transportation), 62, 4.

Lighthill, M. H., and Whitham, G. B., 1955. On kinematic waves II: A theory of traffic flow on long, crowded roads. In Proceedings of The Royal Society of London Ser. A 229, 317-345.

Maerivoet, S., 2006. Modelling Traffic on Motorways: State-of-the-Art, Numerical Data Analysis and Dynamic Traffic Assignment, PhD dissertation. Katholieke Universiteit Leuven.

Mannering, F. L., Kilareski, W. P., and Washburn, S. S., 2005. Principles of Highway Engineering and Traffic Analysis. Third Edition. Chapter 5.

Meignan, D., Simonin, O., and Koukam, A., 2007. Simulation and evaluation of urban bus-networks using a multiagent approach. In Simul. Model. Pract. Theory, vol. 15, no. 6, pp. 659–671.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Masutani, O.; Sasaki, H.; Iwasaki, H.; Ando, Y.; Fukazawa, Y. & Honiden, S., 2005. Pheromone model: application to traffic congestion prediction. In proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM, 1171-1172.

McFarlane, D.C., and Bussmann, S., 2000. Developments in Holonic Production Planning and Control. International Journal of Production Planning and Control, 11(6), pp.522-36.

Michael, G., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M., 1999. The Belief-Desire-Intention Model of Agency. Intelligent Agents V: Agents Theories, Architectures, and Languages.

Mun, J.-S., 2007. Traffic Performance Models for Dynamic Traffic Assignment: An Assessment of Existing Models. Transport Reviews, 27(2), pp.231-49.

Newell, G. F., 1995. Mathematical Models for Freely-Flowing Highway Traffic. Journal of Oper. Res. Soc. Amer., 3 (2), pp. 176–186.

Newell, G., 2002. A simplified car-following theory: a lower order model. Transportation Research Part B, 36, pp. 195–205.

Nie, X., and Zhang, H.M., 2005. A Comparative Study of Some Macroscopic Link Models Used in Dynamic Traffic Assignment. Networks and Spatial Economics, 5, pp.89-115.

Nikolai, C., and Madey, G., 2009. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. Journal of Artificial Societies and Social Simulation, 12, (2)2, <http://jasss.soc.surrey.ac.uk/12/2/2.html>.

Olstam J. J. and Tapani, A., 2004. Comparison of car-following models. Technical report, Swedish National Road and Transport Research Institute.

Payne, H. J., 1979. A critical review of a macroscopic freeway model. In Engineering Foundation Conference on Research Directions in Computer Control of Urban Traffic. Editors W. S. Levine, E. Lieberman and J. J. Fearnsides, pp. 251–265.

Philips, J. et al., 2011. PROSA and Delegate MAS in Robotics., 2011. Springer.

Ranjitkar, P., Kawamua, A., and Nakatsuji, T., 2005. Car-following models: An experiment based benchmarking. Journal of the Eastern Asia Society for Transportation Studies, 6, pp. 1582–1596.

Rao, A. S., and Georgeff, M. P., 1995. BDI agents: from theory to practice. In proceedings of the first international conference on Multi agents systems (ICMAS-95), http://www.citeseer.nj.nec.com/rao95bdi.html.

Richards, P.I., 1956. Shockwaves on the highway. Operations Research, 4, pp.42-51.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Roozemond, D. A., 1999. Using autonomous intelligent agents for urban traffic control systems. In Proc. 6th World Congr. Intell. Transp. Syst.,Toronto, ON, Canada.

Ross, D.T., 1985. Applications and Extensions of SADT. Computer, 18(4), pp.25-34.

Rossetti, R. J. F., Bampi, S., Liu, R., and Van Vliet, D., 2000. An agent-based framework for the assessment of drivers' decision-making. In Proc. IEEE Intell. Transp. Syst., Dearborn, MI, pp. 387–392.

Srinivasan, D., and Choy, M. C., 2006. Cooperative multi-agent system for coordinated traffic signal control. In Proc. Inst. Elect. Eng.—Intell. Transp.Syst., vol. 153, no. 1, pp. 41–50.

Tatomir, B., Rothkrantz, LJ., Suson, AC., 2009. Travel time prediction for dynamic routing using ant based control. In: proceedings of the Winter Simulation Conference, pp. 1069–1078.

Theraulaz, G., and Bonabeau, E., 1999. A brief history of stigmergy. Artificial Life, 5(2), pp. 97-116.

Tomas, V. R., and Garcia, L. A., 2005. Agent-based management of non urban road meteorological incidents. In Proc. Multi-Agent Syst. Appl. IV, vol. 3690, pp. 213–222.

Valckenaers, P., and Van Brussel, H., 2005. Holonic Manufacturing Execution Systems. CIRP Annals - Manufacturing Technology, 54(1), pp.427-32.

Van Belle, J. et al., 2012. A Service-Oriented Approach for Holonic Manufacturing Control and Beyond. In T. Borangiu, A. Thomas & D. Trentesaux, eds. Service Orientation in Holonic and Multi-Agent Manufacturing Control. Springer. Ch. 1. pp.1-20.

Van Belle, J. et al., 2011. Bio-Inspired Coordination and Control in Self-Organizing Logistic Execution Systems., 2011.

Van Brussel, H. et al., 1998. Reference architecture for holonic manufacturing systems:

Verstraete, P., et al., 2008. Engineering manufacturing control systems using PROSA and delegate MAS. Int. J. Agent-Oriented Softw. Eng. 2, 1 62-89.

Wahle, J., Bazzan, A. L. C., Klugl, F., and Schreckenberg, M., 2002. The impact of real-time information in a two-route scenario using agent-based simulation. Transp. Res. Part C: Emerging Technol., vol. 10, no. 5/6, pp. 399–417.

Wang, F. Y., 2005. Agent-based control for networked traffic management systems. IEEE Intell. Syst., vol. 20, no. 5, pp. 92–96.

Weyns, D., Holvoet, T., and Helleboogh, A., 2007. Anticipatory vehicle routing using delegate multi-agent systems. In Proc. IEEE Intell. Transp. Syst. Conf., 2007, pp. 87–93.

Wooldridge, M., 2009. An Introduction to Multiagent Systems. 2nd ed. Wiley Publishing.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 288205.

Zheng, J., Suzuki, K., and Fujita, M., 2012. Evaluation of Car-following Models Using Trajectory Data from Real Traffic. Procedia - Social and Behavioral Sciences, Volume 43, p. 356-366.