Context-Driven Dynamic Organizations Applied to Coordinated Monitoring of Traffic Jams

Robrecht Haesevoets, Bart Van Eylen, Danny Weyns,
Alexander Helleboogh, Tom Holvoet, Wouter Joosen

DistriNet, Katholieke Universiteit Leuven
Celestijnenlaan 200 A, B-3001 Leuven, Belgium
{firstname.name}@cs.kuleuven.be

Abstract. Organizations are at the heart of multi-agent systems. To deal with the ongoing dynamics and changes in the system, organizations have to adapt. Typically, agents are responsible to deal with the complexity of organization dynamics. In this paper, we present an approach for context-driven dynamic organizations in which the agent environment takes the burden of managing organization dynamics. Driven by the context, the agent environment manages the evolution of organizations and actively advertises roles to the agents, supporting the necessary collaborations between agents needed in the current context. We introduce a conceptual model for context-driven dynamic organizations and present a software architecture that supports the model in a distributed setting. The proposed approach separates the management of dynamic evolution of organizations from the actual functionality provided by the agents playing roles in the organizations. Separating these concerns makes it easier to understand, design, and manage organizations in multi-agent systems.
We show how we have applied context-driven dynamic organizations in a concrete case of monitoring traffic jams. In this case, camera agents associated with traffic monitoring cameras collaborate in organizations. Depending on the context, camera agents play different roles, with responsibilities ranging from simple measurement to data aggregation. When a traffic jam covers the viewing range of multiple cameras, organizations are dynamically merged, assuring cameras detecting the same traffic jam can collaborate. Vice versa, when a traffic jam dissolves, the organization is dynamically split up. Test results indicate that context-based dynamic organizations is a promising approach to support decentralized traffic monitoring.

1 Introduction

Structuring and managing interactions among agents is a crucial part of the design of any multi-agent system. A typical way to manage these interactions is by means of organizations in which agents play roles [23, 25, 8]. Multi-agent systems are often applied to problems and domains which are very dynamic in nature. To deal with the ongoing dynamics and changes, organizations have to adapt. In many systems the interactions between agents in organizations are imposed or driven by the current context or environment of the system. Depending on the context, agents have to collaborate in organizations and play certain roles. Most of the existing work on organizations defines roles and organizations at the level of agents [12, 13, 28, 10]. As such, agents have a dual role: on the one hand agents play roles providing the associated functionality in the organization, on the other hand agents are responsible to set up and manage organizations, and deal with the complexity of organization dynamics.

In this paper, we propose an approach called *context-driven dynamic organizations* that considers an organization as a first-class abstraction which is explicitly supported by the agent environment.[1] In particular, the agent environment takes the burden of managing organizations and their dynamics. Driven

[1] In line with [35], we consider the agent environment as an explicit building block in a multi-agent system that encapsulates its own clear-cut responsibilities, irrespective of the agents. Note that the agent environment should not be confused with the *environment* in which the system is deployed, i.e., the part of the external world with which the multi-agent system interact, and in which the effects of the system will be observed and evaluated [20].

by the context, the agent environment manages the evolution of organizations and actively advertises roles to agents, supporting the necessary collaborations between agents needed in the current context.

The proposed approach separates the management of dynamic evolution of organizations from the actual functionality provided by the agents playing roles in the organizations. Separating these concerns makes it easier to understand, design, and manage organizations in multi-agent systems. We present a software architecture that supports our model for context-driven dynamic organizations in a distributed setting. In this architecture the agent environment consists of a group of distributed local agent environments. The local agent environment provides functionality to the agents for perception, action and interaction, and it manages organizations which dynamically evolve according to the current context.

We apply context-driven dynamic organizations to a concrete case. The case covers the coordinated monitoring of traffic jams and clearly shows the need for context-driven dynamic organizations. In this case, camera agents associated with traffic monitoring cameras collaborate in organizations. Depending on the context, camera agents play different roles, with responsibilities ranging from simple measurement to data aggregation. When a traffic jam covers the viewing range of multiple cameras, organizations are dynamically merged, assuring cameras detecting the same traffic jam can collaborate. Vice versa, when a traffic jam dissolves, the organization is dynamically split up. Test results indicate that context-driven dynamic organizations is a promising approach to support decentralized traffic monitoring.

Overview of the paper. Section 2 starts by introducing the concrete case on coordinated monitoring of traffic jams. In Sect. 3 we introduce the conceptual model for context-driven dynamic organizations and in Sect. 4 we present a software architecture that supports this model. Section 5 shows a prototype implementation for the traffic monitoring case, giving an initial validation of our model. Section 6 discusses related work, and in Sect. 7 we draw conclusions.

2 Case Study: Coordinated Monitoring of Traffic Jams

Intelligent transportation systems are a worldwide initiative to exploit information and communication technology in traffic monitoring and control, aiming to improve safety and to reduce vehicle wear, transportation times and fuel consumption [19, 11]. In this case, we focus on monitoring traffic jams on highways with a number of intelligent cameras. Traffic monitoring knows many applications, such as informing drivers about expected travel time delays, driving traffic control applications such as traffic light regulation, collecting data for long term structural decision making, etc.

The monitoring application we consider, consists of a set of intelligent cameras that are distributed evenly along the highway, as shown in Fig. 1. Each camera has a limited viewing range and cameras are placed to get an optimal coverage of the highway with a minimum in overlap. Each camera is able to measure three traffic variables within its viewing range: the current density ($k \approx$ number of vehicles per length unit), intensity ($q \approx$ number of vehicles per time unit) and average speed (u). These three variables can be used to determine the current congestion level and decide whether there is a traffic jam or not [34]. Every camera is equipped with a data processing unit, capable of processing the monitored data, and a communication unit, to communicate with other cameras. The task of the cameras is to detect and monitor traffic jams on the highway. This case will follow an approach in which traffic jams are monitored in a decentralized way, avoiding the bottleneck of a centralized control center.

Because a camera has only a limited viewing range, no single camera will be able to monitor the complete highway or even a single traffic jam. Traffic jams can cover the viewing range of multiple cameras and can dynamically grow and dissolve. When growing or dissolving, traffic jams can enter and leave the viewing range of several cameras. To monitor a traffic jam, data observed by multiple cameras has to be aggregated. Because there is no central point of control, cameras will have to collaborate, and distribute the aggregated data to the necessary traffic signs, controllers, etc.

By default each camera will simply monitor the traffic variables (density, intensity and average speed) of the traffic within its viewing range. When a traffic jam occurs, in the viewing range of a camera, the

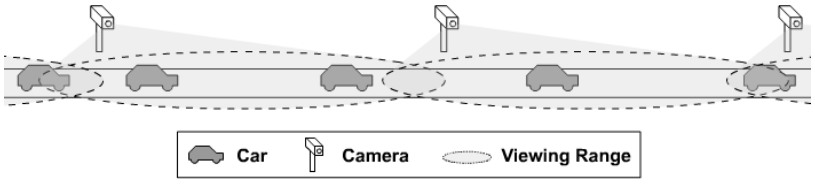


Fig. 1. An example of a highway with traffic cameras.

locally observed variables will pass a certain threshold. The camera will then have to collaborate with other cameras, detecting the same traffic jam. In the collaboration, the data each camera is monitoring is aggregated to get a complete image of the traffic jam. Cameras will enter or leave the collaboration, whenever the traffic jam enters or leaves their viewing range.

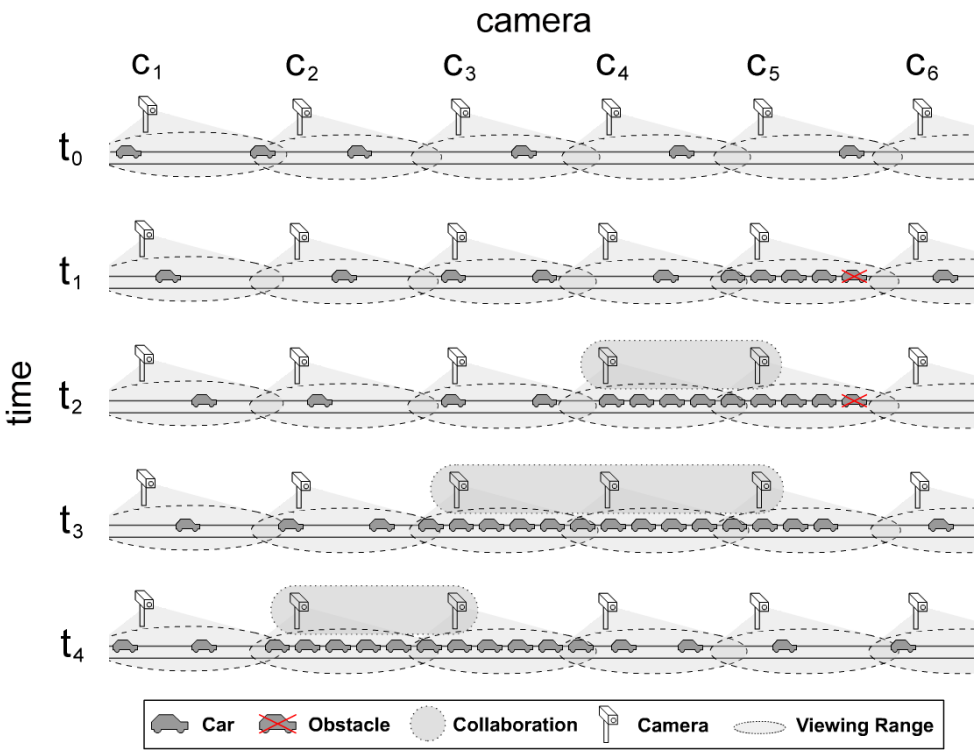


Fig. 2. An example of collaborations between traffic cameras.

An example of such a collaboration is shown in figure 2. At t_0, there is no traffic jam and all cameras monitor the traffic variables. At t_1, an accident occurs in the viewing range of camera c_5 and a traffic jam arises. The traffic jam is only visible in the viewing range of camera c_5 and there is no collaboration necessary among the cameras. The traffic jam, however, starts to grow and at t_2 it has entered the viewing range of camera c_4. Camera c_4 and c_5 start to collaborate because they are now both monitoring the same traffic jam. At t_3, the accident is solved but the traffic jam has further grown and entered the viewing range of camera c_3. Therefore, camera c_3 now participates in the collaboration between camera c_4 and c_5. At t_4, the traffic jam has entered the viewing range of camera c_2 but has dissolved in the viewing range

of camera c_4 and c_5. Camera c_4 and c_5 have stopped collaborating while camera c_2 is collaborating with camera c_3. This example scenario illustrates how the collaboration between the cameras is driven by the context.

The dynamic nature of the traffic phenomena demands for dynamic collaborations betweens the cameras. Cameras will have to collaborate in organizations, which have to evolve dynamically according to the current traffic conditions, which make up the context of the highway. In the following sections, we will show how our conceptual model and architecture for context-driven dynamic organizations can be applied to the case of monitoring traffic jams and offer support for the complex collaborations required between the cameras.

3 Model for Context-Driven Dynamic Organizations

In this section, we present a conceptual model for context-driven dynamic organizations. The complete model is shown in Fig. 3. The model offers support for organizations and roles which dynamically evolve according to the current context. The idea behind the model is to separate the context-driven dynamic evolution of organizations from the actual functionality provided by the agents playing roles in the organizations. This separation of concerns supports a system in which the agent environment provides support for organizations and manages organization dynamics, while the agents provide the actual functionality and behavior required in the organizations.

The main focus of the model is on how existing organizations dynamically evolve. How organization are set up and what the relations are between roles is currently outside the scope of the model. Relations between roles, however, is a well researched topic, as we will cover in Sect. 6, and we consider it future work to add explicit support for this to our model.

Section 3.1 starts with explaining the conceptual model, and how the concepts allow to separate the dynamics of organizations from the actual functionality and behavior. After the introduction of each concept, it is mapped to the traffic monitoring case, introduced in Sect. 2. In Sect. 3.2 we illustrate how organizations can dynamically evolve, driven by the context, and finally in Sect. 3.3 we cover how the context itself is represented in the model.

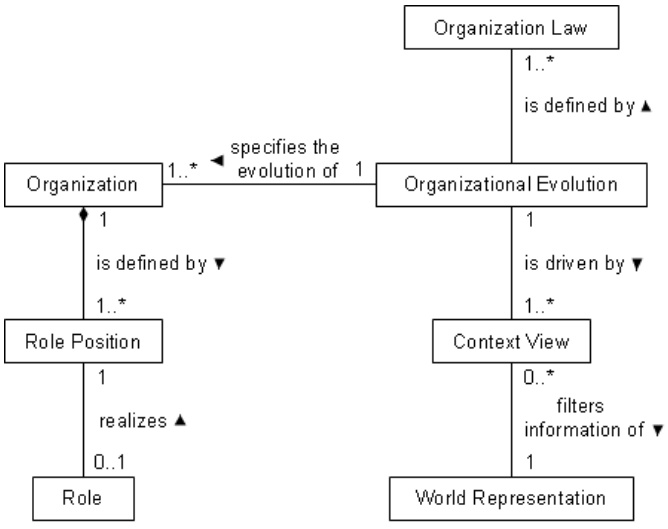


Fig. 3. Conceptual model for context-driven dynamic organizations.

3.1 Organizations, Role Positions and Roles

An organization (`Organization`) is defined by a set of related role positions (`Role Position`). A role position is very similar to a job opening in a company. It represents the specification of a coherent part of functionality required in an organization, but it can also include a part of the infrastructure needed to realize the required functionality. A role (`Role`) realizes the actual functionality specified by a role position. To collaborate with each other, agents can play one or multiple roles in one or more organizations.

The abstraction of organization allows to address a group of collaborating agents as one entity without knowing the individual agents. The abstraction of role position separates the dynamics of an organization, driven by the current context, from the actual functionality of the organization, provided by the agents playing the corresponding roles. It also allows agents to transparently swap roles within or between organizations, while leaving the decision to do so to the agents.

The difference between a role position and a role is clarified by the following analogy. A hospital has a role position for a surgeon. This role position represents the specification of the job as a surgeon, but also includes the necessary infrastructure, such as office with a telephone and an internet connection, and an operating room, needed by the doctor playing the role of surgeon. It is clear that the role position of surgeon can exist without a doctor actually playing the role of surgeon, and that different doctors with the necessary qualifications and approval can decide whether or not to play the role of surgeon. The hospital management, however, can decide whether or not to open a new role position for surgeon or to close an exiting one, depending on the current demand of patients. The notion of role positions allows the hospital management to define and change the organizational structure, independent of the doctors providing the actual service of a surgeon.

Organizations, roles positions and roles in the traffic monitoring case. In the traffic monitoring case, a software agent is deployed on every camera. The agent is capable of playing roles and will further be referred to as camera agent. A congestion monitoring organization allows camera agents to collaborate, in order to detect and monitor traffic jams. When a camera agent enters the system, it is automatically added to a new a congestion monitoring organization. Within the congestion monitoring organization, there are three kinds of role positions possible: data observer (`Data Observer`), data pusher (`Data Pusher`) and data aggregator (`Data Aggregator`). The data observer role is responsible for monitoring the three traffic variables (density, intensity and average speed) and deciding whether the congestion level is high enough to be considered a traffic jam. The data pusher role is responsible for pushing the observed data to the data aggregator role, which in turn is responsible for aggregating the data and distributing it to the necessary clients.

3.2 Context-Driven Evolution of Organizations

The evolution of organizations and their role positions is context-driven. Organizational evolution (`Organizational Evolution`) uses a view on the current context (`Context view`) as input to evolve an organization and its role positions. Context consists of information such as existing role positions and their associated roles, information on external resources probed by sensors, or data related to physical entities. Organizations can evolve by splitting up or merging together, regrouping the agents to support the necessary collaborations between agents, needed in the current context. Role positions are evolved by opening new positions or closing existing ones.

Organizational evolution (`Organizational Evolution`) uses a set of laws (`Organization Laws`), which define the way organizations and role positions should evolve given the current context. Currently the model supports three kinds of laws: split laws (`Split Law`), merge laws (`Merge Law`) and role positioning laws (`Role Positioning Law`).

Role positioning laws define whether existing role positions should be closed or whether new positions should be opened. Merge laws define whether two or more organizations should be merged into one

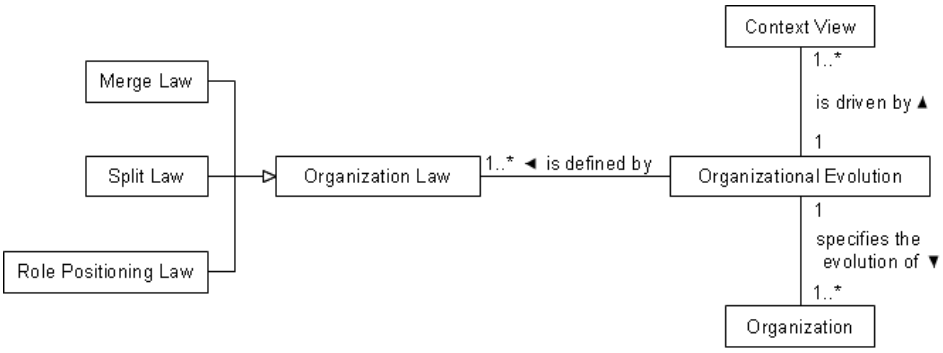


Fig. 4. Different kinds laws used by the organizational evolution.

organization, and split laws define whether an organization should be split up. Organizational evolution is the evolution of organizations according to these laws. Whenever something changes in the context, the laws are re-evaluated and organizations and role positions are updated accordingly.

The model assumes that laws are applied atomically and that laws can be executed in any order when multiple laws apply. Figure 5 shows a statechart representation of an example of a possible evolution of organizations and role positions. In the example, the current configuration of organizations and role positions, and the current context is represented as a state. Whenever something changes in the context or when laws are applied, a new state is reached. In the example the context in state S_0 is changed and a new state S_1 is reached. Because the context is changed, laws are re-evaluated and applied in an atomic and unordered manner. In the example law_B is first applied and a new state S_2 is reached, representing an updated configuration of organizations and role positions and an updated context. Next law_A is applied and a new state S_3 is reached, again with an updated configuration and context.

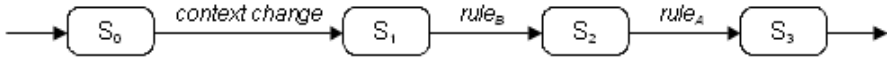


Fig. 5. Statechart representation of an example of organizational evolution.

Context-Driven Evolution of Organizations in the Traffic Monitoring Case. When a camera agent enters the system, the agent is automatically added to a new congestion monitoring organization. The further life of the organization is guided by the organizational evolution. Camera agents can not join or leave new organizations, but the existing organizations can be merged together or split up, regrouping the agents according to the current context. Role positions can be dynamically opened or closed.

Organizational evolution is defined by one merge law, one split law and one role positioning law per type of role position. The merge law states that congestion monitoring organizations should be merged when they are neighbouring organizations in space and when they are both monitoring a traffic jam. The split law states that camera agents in an organization with no traffic jam in their viewing range, should be split up in a separate organization. The role positioning laws state that within a congestion monitoring organization there is always a data observer role position available for every agent in the organization, which camera agents are supposed to play by default. The role positions for data pusher and data aggregator are only available when the organization is currently monitoring a traffic jam and data needs to be aggregated. If the organization is monitoring a traffic jam, the role position of data

pusher will be available for every agent in the organization, but only one position will be available for the data aggregator role.

According to these laws, when there is no traffic jam, organizations will consist of single camera agents. The camera agents in these organizations will then play the default role of data observer, to detect whether a traffic jam arises. When a traffic jam is detected, the organization will grow along with the traffic jam in order to fully monitor it. Organizations grow by merging with neighbouring organizations, detecting the same traffic jam, as defined by the merge law. When the traffic jam dissolves or leaves the viewing range of a camera, the organization is split up again, as defined by the split law. This way, only camera agents with the traffic jam in the viewing range of their camera will be grouped in one organization.

Figure 6, shows a reprise of the example in Fig. 2, but now with the model applied to it. At t_0, there is no traffic jam and all camera agents are member of a separate congestion monitoring organization. At t_1, an accident occurs in the viewing range of camera c_5 and a traffic jam originates. The traffic jam starts to grow and at t_2 it has entered the viewing range of camera c_4. The organizations of camera agents c_4 and c_5 are merged together because they are both monitoring the same traffic jam. At t_3, the accident is solved but the traffic jam has further grown into the viewing range of camera c_3. The organization of camera agent c_3 is therefore merged with the organization of camera agents c_4 and c_5. At t_4, the traffic jam has grown into the viewing range of camera c_2 but has left the viewing range of camera c_4 and c_5. The organization is split up, camera agents c_4 and c_5 are in separate organizations and the organizations of camera agents c_2 and c_3 are merged together.

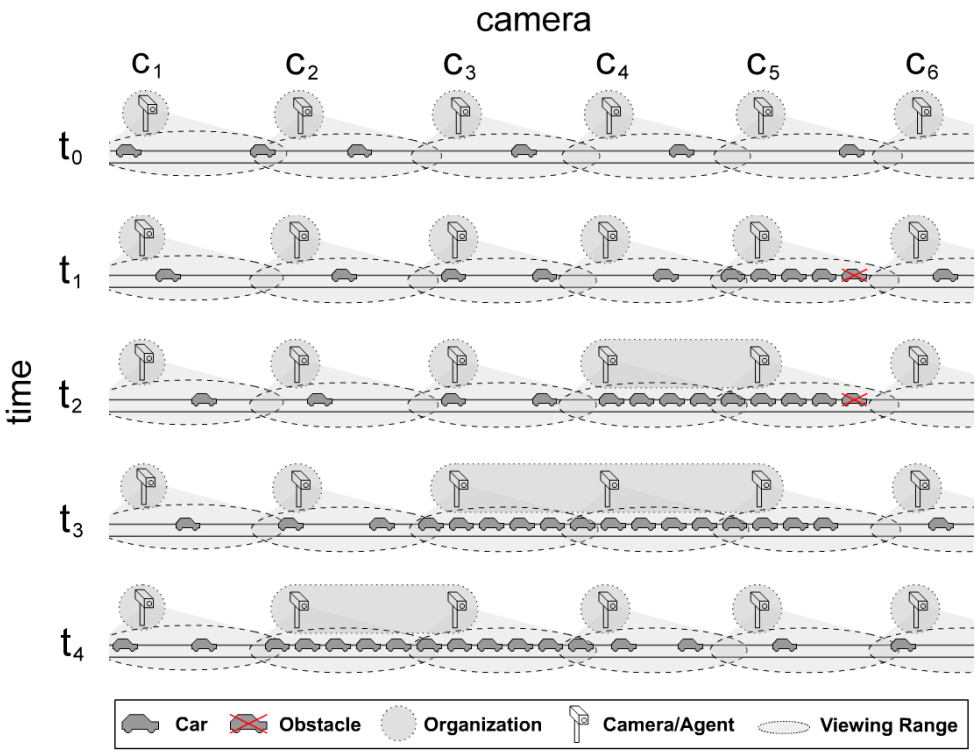


Fig. 6. An example of congestion monitoring organizations.

3.3 World Representation and Context View

As mentioned before, context is an important concept in our model, it is used as input for the organizational evolution. World representation (`World Representation`) is an abstraction of the current state or context of the world. World representation can contain information on the state of the real world, such as existing entities or input readings from sensors, as well as information on the state of the virtual world, such as existing organizations and role positions. However, world representation contains a lot of information which is often not relevant or needed and is therefore not accessed directly, but through a context view (`Context View`). A context view represents a specific view on the world representation, only focussing on relevant information. Organizational evolution can have one or more of these context views, e.g., each focussing on information of a specific organization.

World representation and context view in the traffic monitoring case. In the traffic monitoring case the world consist of a highway with cars on it, a set of distributed cameras, their organizations and their roles. The organizational evolution will have a context view on each organization. The context view on an organization consists among other things of the open and implemented role positions, the current congestion level in the viewing range of individual cameras and information on neighbouring organizations it can be merged with.

4 Supporting Architecture

In this section, we present a high-level software architecture that supports the presented model for context-driven dynamic organizations. Figure 7 shows the collaborating components view of the software architecture that is deployed on each node or intelligent camera. The collaborating components view shows the agent system as a set of interacting components that use a set of shared data repositories to realize the required system functionalities. The main software components are `Local Agent Environment` and `Agent`. The local agent environment enables agents to coordinate their behavior [35]. It provides functionality to the agents for perception, action and interaction, but it also provides organizations, which it dynamically evolves according to the current context. The local agent environment is connected to the external world which contains the given hardware and software with which the software system interacts.

The evolution of organizations, or organizational evolution, is managed by the `Organization Manager` components. The world representation is maintained by the `Local World Representation` and the `Local Orgs State` repositories. The local world representation repository contains a representation of the external world. This representation includes data about external resources probed by sensors, and data related to entities, associated with the agents in the system such as the spatial position of cameras. The local orgs state repository contains state about the current organizations, such as agent-role pairs. The data of the local world representation is kept up to date by the `Synchronization` component, which maintains the representation of relevant local resources in the external world and synchronizes its state with synchronization components on other nodes, using the `Inter Node Communication Infrastructure`. The local orgs state repository also has a synchronization component which synchronizes its state with other nodes.

The `Organization Controller` can control the way the local world representation and the local orgs state repositories are synchronized with the repositories in other nodes. This enables the organization controller to decide which data should be send to other nodes and which data should be retrieved to get an up to date version. Based on the current context derived from the local world representation and the local orgs state repository, the organization controller can initiate an organization. It therefore opens a number of `Open Role Positions`. The `Role Position Pool` `advertises` the open role positions. When an agent `selects` an open role position, the organization controller instantiates a `Role Position` that is attached to a corresponding `Role` played by the agent. The organization

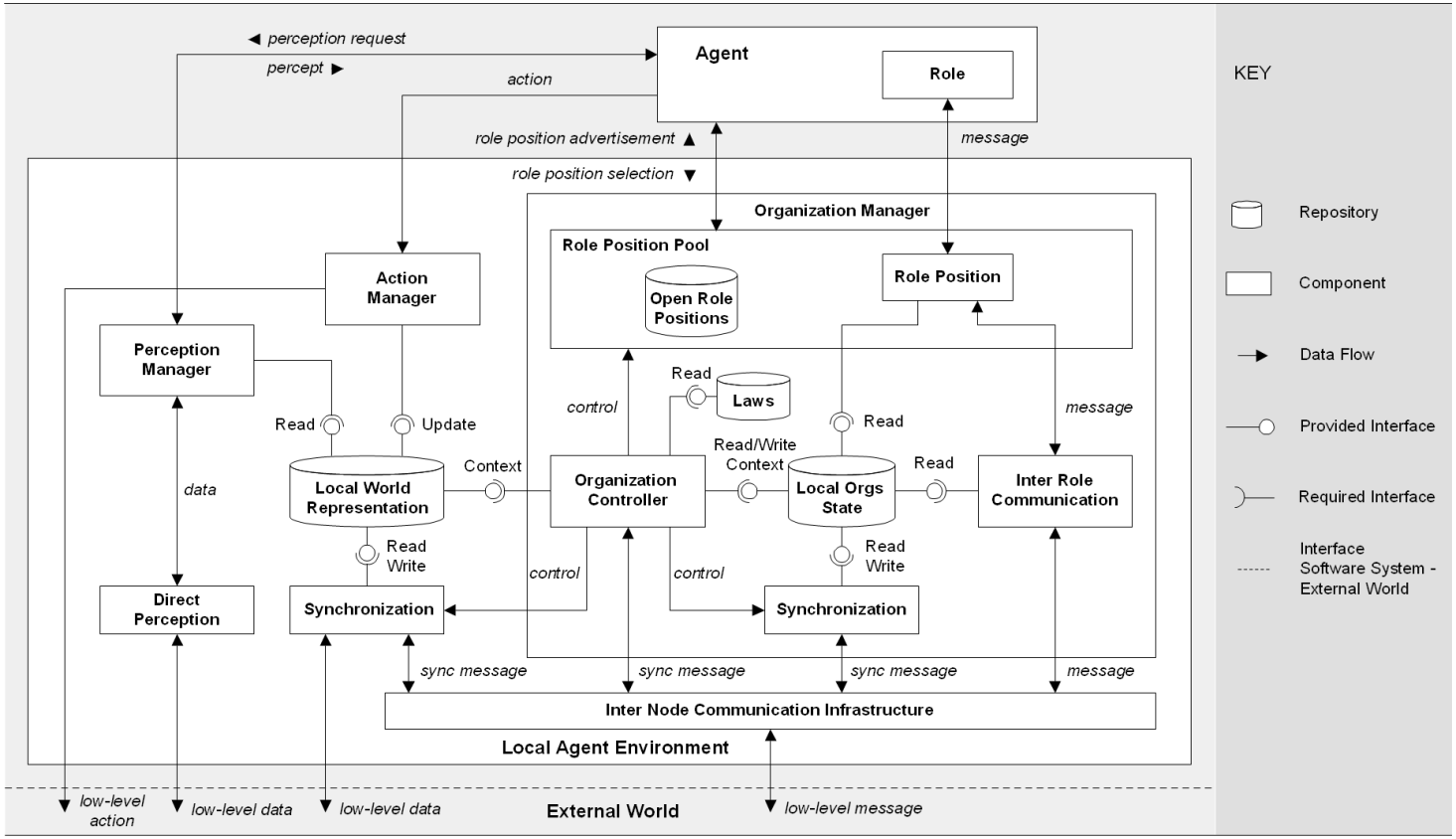


Fig. 7. Collaborating Components view of the software architecture for context-driven dynamic organizations.

controller maintains the state about the current organizations in the local orgs state repository. The organization controller monitors changes in the local world representation as well as changes in the local orgs state. Based on these observations, an organization law in the `Laws` repository can be triggered, inducing a change in an organization. If such a change occurs, the organization controller may interact with controllers on other nodes and subsequently adapt the set of role positions accordingly (positions may be closed, or new positions may be opened) which in turn will inform the roles of the organization.

Agents that play a role in an organization can interact by sending `messages` to one another through their role positions. The `Inter Role Communication` component uses the local orgs state repository to translate the role position to the correct node address. The agent itself can also read the local orgs state repository through its role positions. This enables the agent to get information on the organizations it is involved in, such as other role positions it can interact with. The `Perception Manager` [36] enables an agent to perceive the external world according to a particular `perception request`, resulting in a `percept`. It allows the agent to perceive the external world in an indirect manner through the local world representation or in a direct manner through the `Direct Perception` component. Direct perception is used to access data in the external world, which is to big to be kept up to date in a local repository, such as video images from a camera sensor. The `Action Manager` enables an agent to perform actions in the external world, such as tilting the camera or adjusting the camera's focus, and to update the local world representation with high-level or interpreted data, such as the current congestion level in the camera's viewing range.

Figure 8 shows an excerpt of the software architecture applied to the traffic monitoring application. The figure reflects the organization structure of cameras c_4, c_5, and c_6 at time t_2 in Fig. 6. In the depicted situation, the camera agents of c_4 and c_5 form an organization, monitoring a traffic jam. Since there is no traffic jam within the viewing range of c_6, the camera agent of c_6 makes up an organization on its own.

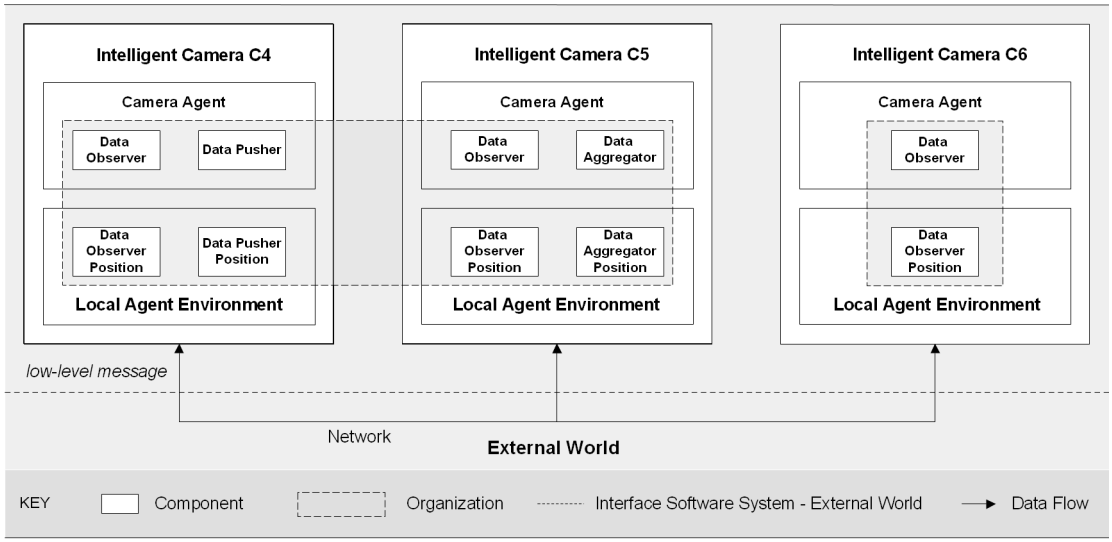


Fig. 8. Software architecture applied to the traffic monitoring application.

5 Prototype and Evaluation

The goal of the prototype and evaluation is twofold. First, it is used as a validation for the model for context-driven dynamic organizations, introduced in Sect. 3, applied to the traffic monitoring case. Sec-

ond, it is used to give an initial indication of the communication cost to support dynamic evolution of organizations. This section will first explain the setup used for the validation and the assumptions the prototype is built on, before explaining the concrete experiments.

The prototype is built on a cellular automaton based traffic simulator [5]. It allows for the discrete simulation of individual vehicles, based on different selectable driver models. We extended the simulator with the notion of cameras, for measuring density, intensity and average speed. On top of the extended simulator, we built a prototype implementation of the architecture, presented in Sect. 4, explicitly supporting the abstractions provided by the model, introduced in Sect. 3. It is important to note that we have currently made abstraction from distribution issues.

In our experiments we modelled a straight highway with a length of 4500 metres, one lane and one direction. Thirty cameras are equally distributed along the highway. First, we examined the dynamic evolution of organizations in four different scenarios. Second, we used the same scenarios to give an indication of the number of messages needed for the dynamic evolution of organizations.

The results of the experiment are presented by a set of snapshots, an example of such a snapshot is given in Fig. 9(a), the key is given in Fig. 9(d). The top of the Fig. 9(a) is a space time plot. It shows the trajectory of each individual vehicle through time. Time t in the plot indicates the time step on which the snapshot was taken. Values on the time axis, smaller than t, happened at a time in the past relative to the snapshot. This way the space time plot shows the evolution of all the different vehicle trajectories in a fixed sized time frame. The color of each point in the graph represents the speed of a vehicle at the specified time and position. Green indicates the maximum speed, red indicates the lowest speed. Beneath the space time plot is a snapshot of the situation on the highway at time step t. It is aligned with the space axis and therefore maps directly to the different trajectories in the space time plot. At the bottom another space time plot is given, showing the state of the congestion monitoring organizations. The different columns in the plot represent the thirty cameras. The color of each column represents the organizational situation of each camera. Green means that at that specific time the camera sees no congestion and sits all by itself in its organization playing the role of data observer. Blue means that the camera sees congestion but still sits alone in its organization, playing both the role of data pusher and data aggregator. Red means that the camera observes congestion and forms an organization with neighbouring cameras if they also have a red color. This way, adjacent red columns represent one organization.

Free flow traffic: In a first scenario we monitored free flow traffic, which constitutes of a continuous stream of vehicles traversing the highway. A snapshot of the situation is given in Fig. 9(a), which, as expected, shows that there is no congestion and all cameras are playing the role of data observer in their own organization.

Obstacle blocking highway: In a second scenario we simulated an obstacle in the middle of the highway to initiate a traffic jam. The evolution of this scenario is shown in Fig. 9(b), 9(c) and 10(a). In Fig. 9(b) we see in the space time plot at the top, that a traffic jam has emerged and is propagating backwards over the highway. The snapshot of the highway itself supports this finding. If we look at the bottom of the figure we see that the camera monitoring the congested part of the highway, is now also playing the role of data pusher and data aggregator, indicated by the blue color. Figure 9(c) and 10(a) show the further evolution of the traffic jam. Different adjacent columns have a red color indicating that these cameras are observing congestion and all form one organization. As expected, cameras detect the congestion and over time evolve into one large organization along the congested part of the highway.

Obstacle removed: In a third scenario we removed the obstacle. The traffic jam slowly dissolved in the front, while propagating backwards. The organization formed in the previous scenario should adapt itself accordingly. Figure 10(b) and 10(c) show the outcome of this scenario. The traffic jam slowly dissolves at the front while propagating backwards. As expected, cameras in the front of the organization are split

up in separate organizations, while the organizations of cameras in the back are merged in one large organization.

New obstacle introduced: Figure 10(d) shows the fourth and final scenario, in which after the initial obstacle was removed, a new obstacle is introduced at the same location. The initial traffic jam is still propagating backwards while a new traffic jam arises in the middle. This leads to two different organizations, which both evolve over time each following a traffic jam.

Communication costs: Figure 11 shows the communication overhead throughout the four different scenarios. The counted messages are solely messages needed to dynamically evolve organizations. We make the following two observations. First, the larger an organization is the more overhead it takes to merge it, split it or change role positions in it. This can be seen in the second scenario in which a single organization grows larger and larger together with an increase in message overhead. When the organization splits up in smaller organizations in the third scenario, the number of needed messages also starts to drop. Second, the message overhead depends on the number of organizations. This can be observed in fourth scenario, in which the overhead starts to rise again because a second organization has formed.

6 Related Work

We focus our discussion of related work on organizational concepts in multi-agent systems and on traffic monitoring techniques.

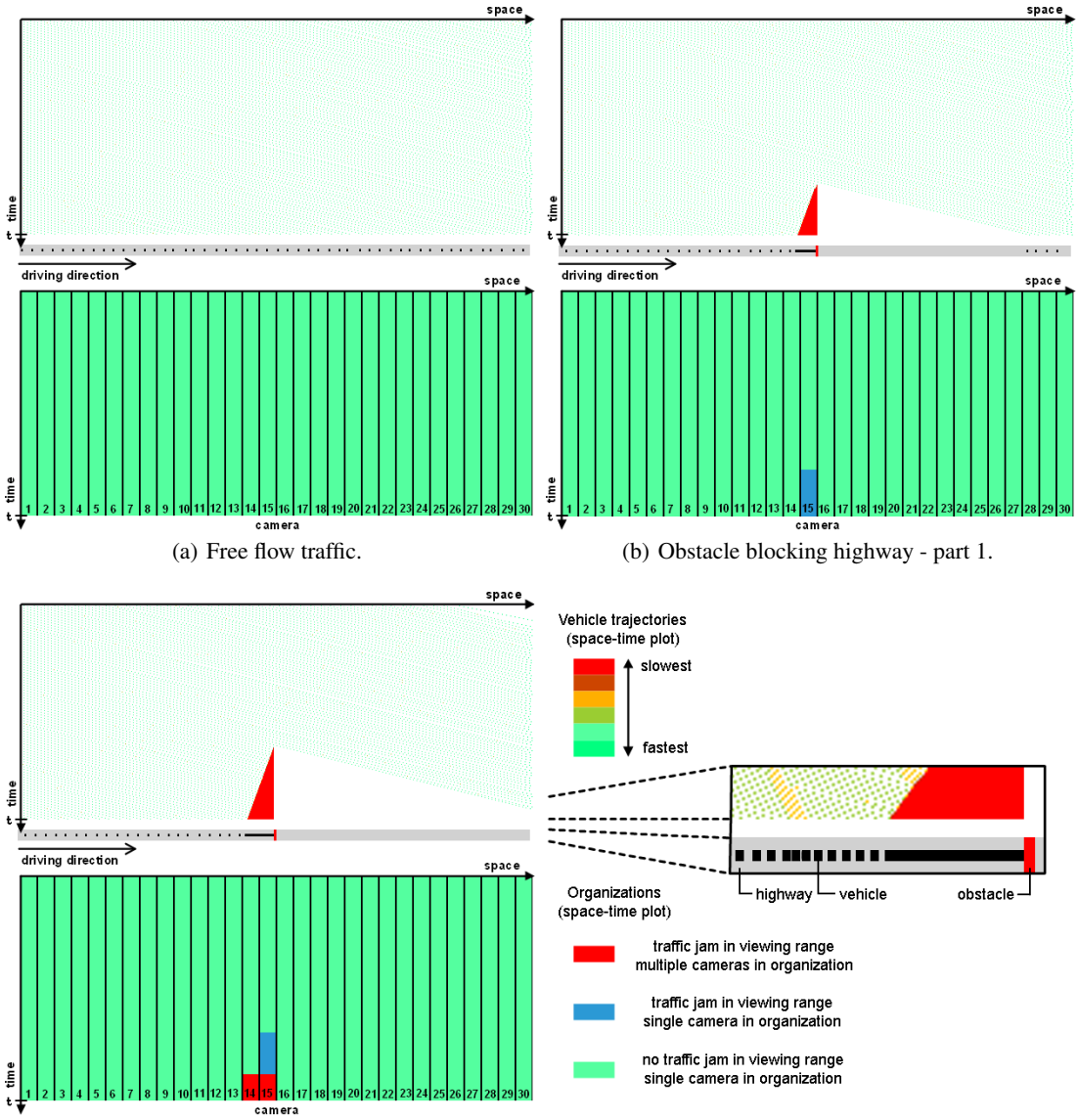
6.1 Organizations and Multi-Agent Systems

We focus on two areas of related work with respect to multi-agent systems research: (1) representation of context information by means of an explicit environment model and (2) roles and organizations in multi-agent systems.

Research on environments in multi-agent systems devotes a lot of attention to representing context information of the external world in an explicit manner, i.e. by means of an explicit model of the environment. For example, to present relevant information of the physical world to agents, [6] introduces a *cognitive middle layer* in the environment that employs a shared ontology to present environmental information to the agents. MMASS [2] introduces a *multi-layered* model of the environment, with each layer an explicit representation of a particular spatial or conceptual structure of the real environment. ELMS [27] is an environment description language with explicit support for specifying perception and interaction of cognitive agents. The main difference between these approaches and our approach is that we employ the environment to support organizations. As such, the context information used in our approach exceeds a representation of the external world, and includes information about the organizational setting of the system.

Roles, organizations and groups are recognized to be valuable abstractions for building multi-agent systems [21, 9, 32, 14]. For example, [26] analyzes role changes in dynamic environments, distinguishing between two categories of changes: dynamic activation of roles and dynamic classification of roles. In [7], the authors define relations between agents and roles, e.g., the way an agent *takes up* a role and *enacts* it. Possible relations between roles and agents and architectural and functional changes that an agent must undergo when it enters an open agent system, are discussed.

An overview of the different types of organizations and their characteristics that can be found in the literature, is given in [18]. The focus of our paper is the context-driven dynamic evolution of organizations. We therefore compare it with related research from the perspective of how organizational evolution is supported. AGR (Agent Group Role) [12] is a generic meta-model of multi-agent systems in which



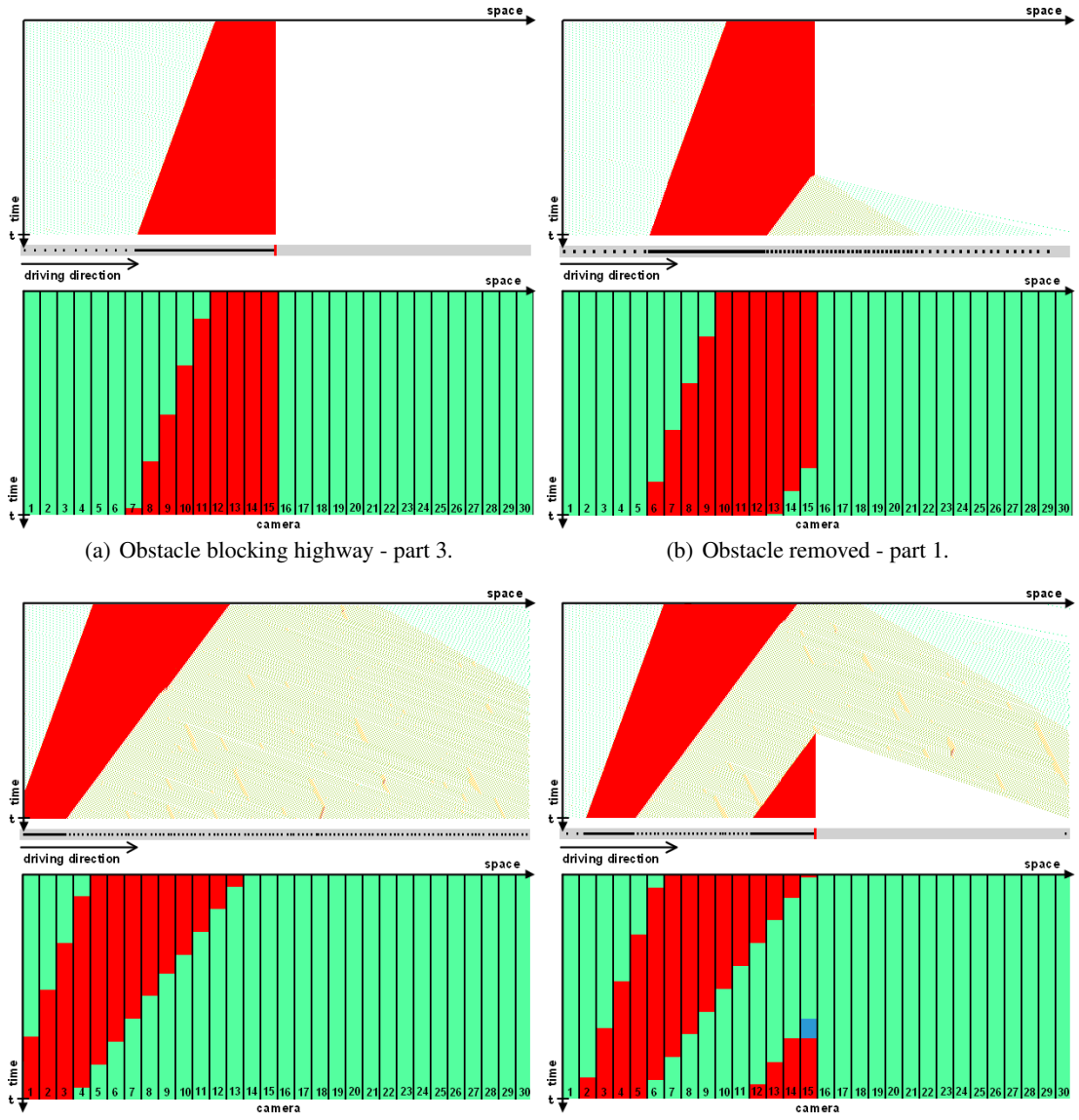
(a) Free flow traffic.

(b) Obstacle blocking highway - part 1.

(c) Obstacle blocking highway - part 2.

(d) Key

Fig. 9. Validation scenarios, part a.



(a) Obstacle blocking highway - part 3.

(b) Obstacle removed - part 1.

(c) Obstacle removed - part 2.

(d) New obstacle introduced.

Fig. 10. Validation scenarios, part b.

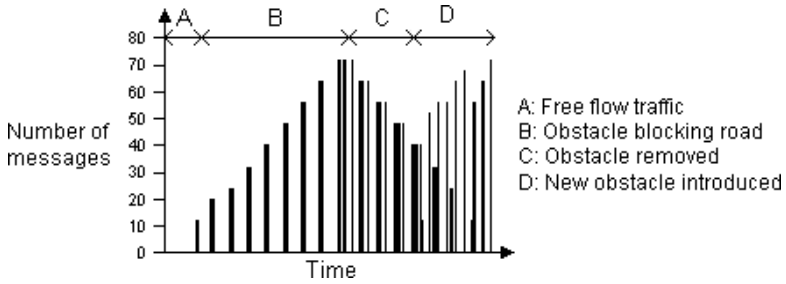


Fig. 11. Message overhead throughout the four different scenarios.

agents, playing roles, are organized into groups. Groups can be seen as our concept of organization. Agents can join or leave a group by communicating with a special agent playing the role of group manager. This role, which is automatically awarded to the group creator, has the responsibility for handling requests for group admission or role requests. It can also revoke roles or group membership. In [13] the AGR model is extended to AGRE with the E standing for environment. Groups are now grouped together into worlds. These worlds offer primitives for agents to join a particular group and to play a particular role. In both models an agent-centric perspective is taken on the dynamic evolution of groups. It is the responsibility of the agents themselves to decide which group they want to join or leave and which role they want to play. This differs from our approach in which the dynamic evolution of organizations is actively managed and driven by the agent environment instead of the agents themselves. The AGR(E) concepts also offer no explicit support to model evolutional changes at the inter-organization level. It is for example not clear how to model that two or more organizations should merge or split in function of the context of the agent system. Our model on the other hand allows to model inter-organizational evolution by means of the first-class concepts of organizational evolution and organization laws.

TuCSoN [29] offers programmable tuplespaces that encapsulate coordination rules between roles. It enables agents to interact at a higher level of abstraction, and in a way that is tailored to the their needs. In [28], TuCSoN is extended to support the description and enacting of organizational models. A TuCSoN node also serves as an organization node, hosting tuple centres as coordination artefacts/services available to agents. In order to access the tuple centres hosted by a node, an agent must join the organization and choose a role to play. As is the case with AGRE, this conforms to an agent-centric approach. Each node also has a special tuple centre that explicitly hosts the description of the organization, represented by that node. It contains, for example, dynamic information about the current set of roles and related relationships (agent-role and inter-role). These organizational settings can be dynamically inspected and changed by agents, by suitably reading and modifying the tuple contents. This allows agents, for example, to add or delete roles from the organization. However, it is only possible to make changes on the intra-organizational level this way. Reorganizations on the inter-organizational level are not supported.

In [10], reorganization issues in agent societies are discussed. It explores how and why organizations change, and how reorganization can be done dynamically, with minimal interference from the system designer (i.e., by the system itself). A classification is given of reorganization situations, based on the focus of the reorganization, the authority to modify the organization, and how reorganization decisions are taken. With respect to the latter, two possible approaches are identified. First, the decision making to change the organization could be the responsibility of one role in the organization. This corresponds to a master/slave relationship between agents acting at the different levels of autonomy and is called role-based control. Second, all or some roles are collectively responsible for a change decision. Changes are then achieved by collaboration or consensus among the agents. This is called shared control. This does not cover our model, in which it is the agent environment that has the authority and decision making power to change organizations.

6.2 Traffic Monitoring

Traffic monitoring is an extensively studied field of research. We discuss a number of representative approaches used for vehicle detection by means of image analysis and multi-sensor fusion.

A lot of research has been done on vision-based traffic surveillance. Many algorithms and processing techniques exist for vehicle detection and/or recognition from single or multiple frames obtained from a camera [3, 16, 22]. [30] describes an approach to identify vehicles in images of congested traffic. They rely on the strong shadows present under each vehicle to detect and localize vehicles. [24] proposes a tracking system, which processes traffic video streams, to track vehicles and to classify these vehicles into three classes, i.e. (1) sedans, (2) semi and (3) truck, SUV or van.

In the traffic monitoring research community, there is an increasing interest in combining footage from multiple (image) sensors to improve the analysis of a traffic situation. For example, [15] uses the

imagery of three cameras, to track the trajectory of passing vehicles. [4] presents a color stereo vision system to extract 3D edges of an observed obstacle. [31] describes an approach, to detect traffic incidents, that relies on different types of sensors: i.e., image sensors and supersonic wave sensors. This allows more accurate detection of traffic incidents.

Support exists to underpin multi-sensor fusion. AROCCAM [33], is a software framework, to design and implement data fusion applications. It provides support for unsynchronized sensors and delayed observations. Other approaches for multi-sensor fusion are [17, 1]. In contrast to our approach, these approaches provide no explicit support to enable sensors to play different roles, or to form organizations between sensors in a context-driven, dynamic manner.

7 Conclusion and Future Work

In this paper, we presented an approach for context-driven dynamic organizations. Contrary to most existing approaches, in which agents have the dual responsibility of playing roles, providing the associated functionality in the organization, and managing organization dynamics, in context-driven dynamic organizations the agent environment takes the burden of managing organization dynamics. Separating the management of dynamic evolution of organizations from the actual functionality, provided by the agents playing roles in the organizations, improves understandability, and enables better reuse, making the design and management of organizations in multi-agent systems easier.

The model is built around four key abstractions: organization, role position, role and context view. Organizations represent dynamic groups of roles played by agents, collaborating to achieve a collective goal. Each role realizes the actual functionalities specified by an associated role position. The dynamical aspects of the organizations are context-driven. The agent environment uses the context as input to manage the dynamic evolution of organizations. Organizations evolve by adapting available role positions, splitting up and merging together, regrouping the agents to support the necessary collaborations between agents needed in the current context.

We presented a high-level software architecture that supports the model for context-driven dynamic organizations and applied it to a traffic monitoring case. With an initial prototype we validated the model and implemented the presented architecture. We also gave an indication about the communication overhead for dynamically evolving organizations, which showed that the communication cost is proportional to the size and number of organizations. Test results indicate that context-based dynamic organizations is a promising approach to support decentralized traffic monitoring.

Future work consists of formalizing the model for context-driven dynamic organizations, investigating the feasibility of introducing explicit support for interaction protocols and relations between roles in the model, and expanding the software architecture to offer middleware support for the presented abstractions.

References

1. Iyad Abuhadrous, Fawzi Nashashibi, and Claude Laurgeau. Multi-Sensor Fusion (GPS, IMU, Odometers) for Precise Land Vehicle Localisation Using RTMAPS. In *11th International Conference on Advanced Robotics (ICAR 2003)*, 2003.
2. S. Bandini, S. Manzoni, and G. Vizzari. A Spatially Dependent Communiction Model for Ubiquitous Systems. In D. Weyns, V. Parunak, and F. Michel, editors, *First International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, New York, NY, USA, 2005. Springer-Verlag.
3. A. Bensrhair, M. Bertozzi, A. Broggi, P. Miché, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. In *4th International IEEE Conference on Intelligent Transportation Systems, ITSC'01*, pages 207–212, Oakland, USA, August 2001.
4. I. Cabani, G. Toulminet, and A. Bensrhair. A color stereo vision system for extraction of 3d edges of obstacle. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*, pages 307–312, 2006.

5. Cellular Automaton Traffic Simulators. http://rcswww.urz.tu-dresden.de/~helbing/RoadApplet/.
6. P. Chang, K. Chen, Y. Chien, E. Kao, and V. Soo. From Reality to Mind: A Cognitive Middle Layer of Environment Concepts for Believable Agents. In D. Weyns, V. Parunak, and F. Michel, editors, *First International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, New York, NY, USA, 2005. Springer-Verlag.
7. M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 489–496, 2003.
8. Y. Demazeau. Multi-Agent Systems Methodology. In *2nd Franco-Mexican School on Cooperative and Distributed Systems, LAFMI 2003, http://lafmi.lania.mx/escuelas/esd03/ponencias/Demazeau.pdf*, 8/2006.
9. Y Demazeau and A.C. Rocha Costa. Populations and organizations in open multi-agent systems. In *Proceedings of the 1st National Symposium on Parallel and Distributed AI*, 1996.
10. V. Dignum, F. Dignum, and L. Sonenberg. Towards Dynamic Reorganization of Agent Societies. *Proceedings of Workshop on Coordination in Emergent Agent Societies at ECAI*, 2004.
11. ERTICO: Intelligent Transportation Systems for Europe. http://www.ertico.com/.
12. J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 128, 1998.
13. J. Ferber, F. Michel, and J. Baez. AGRE: Integrating environments with organizations. In D. Weyns, V. Parunak, and F. Michel, editors, *First International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, New York, NY, USA, 2005. Springer-Verlag.
14. L. Gasser. Organizations in multi-agent systems. In *10th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-2001)*, 2001.
15. Y. Goya, T. Chateau, L. Malaterre, and L. Trassoudaine. Vehicle trajectories evaluation by static video sensors. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*, pages 864–869, Toronto, Canada, September 2006.
16. S. Gupte, O. Masoud, and N. P. Papanikolopoulos. Detection and classification of vehicles. In *IEEE Trans. on Intelligent Transportation Systems*, volume 3, pages 37–47, March 2002.
17. Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The location stack: A layered model for location in ubiquitous computing. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, pages 22–28, Callicoon, NY, June 2002. IEEE Computer Society Press.
18. Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.*, 19(4):281–316, 2004.
19. ITS America: Intelligent Transportation Society of America. http://www.itsa.org/.
20. M. Jackson. The Meaning of Requirements. *Annals of Software Engineering, Special Issue on Software Requirements Engineering*, 3:5–22, 1997.
21. Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.
22. Y.K. Jung, K.W. Lee, and Y.S. Ho. Content-based event retrieval using semantic scene interpretation for automated traffic surveillance. *ITS*, 2(3):151–163, September 2001.
23. E. Kendall. Role modeling for agent system analysis, design, and implementation. *IEEE Concurrency*, 8(2):34–41, 2000.
24. B. Morris and M. Trivedi. Robust classification and tracking of vehicles in traffic video streams. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*, pages 1078–1083, Toronto, Canada, September 2006.
25. J. Odell, H. V. D. Parunak, and M. Fleischer. The Role of Roles. *Journal of Object Technology*, 2(1):39–51, 2003.
26. James Odell, H. Van Dyke Parunak, Sven Brueckner, and John A. Sauter. Changing roles: Dynamic role assignment. *Journal of Object Technology*, 2(5):77–86, 2003.
27. F. Okuyama, R. H. Bordini, and A. C. da Rocha Costa. An Environment Description Language for Multiagent Simulation. In D. Weyns, V. Parunak, and F. Michel, editors, *First International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, NY, USA, 2005.
28. A. Omicini and A. Ricci. Reasoning about organisation: Shaping the infrastructure. *AI* IA Notizie*, 16(2):7–16, 2003.
29. A. Omicini and F. Zambonelli. TuCSoN: a Coordination Model for Mobile Information Agents. In *Proc. of the 1st Workshop on Innovative Internet Information Systems*, 1998.

30. M. Sadeghi and M. Fathy. A low-cost occlusion handling using a novel feature in congested traffic images. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*, pages 522–527, Toronto, Canada, September 2006.
31. N. Sumiya, K. Familiar, and S. Kamijo. Incident detection system by sensor fusion network employing image sensors and supersonic wave sensors. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 1066–1071, 2006.
32. Milind Tambe, David V. Pynadath, and Nicolas Chauvat. Building dynamic agent organizations in cyberspace. *IEEE Internet Computing*, 4(2):65–73, 2000.
33. Cédric Tessier, Christophe Cariou, Christophe Debain, Frederic Chausse, Roland Chapuis, and Christophe Rousset. A real-time, multi-sensor architecture for fusion of delayed observations: Application to vehicle localisation. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*, 2006.
34. Traffic Congestion. `http://en.wikipedia.org/wiki/Traffic_congestion`.
35. D. Weyns, A. Omicini, and J. Odell. Environment as a First-Class Abstraction in Multiagent Systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.
36. D. Weyns, E. Steegmans, and T. Holvoet. Towards Active Perception in Situated Multi-Agent Systems. *Applied Artificial Intelligence*, 18(9-10):867–883, 2004.