Handling of Deviations from Desired Behaviour in Hybrid Central/Self-Organising Multi-Agent Systems

Yaser Chaaban and Christian Müller-Schloer
Institute of Systems Engineering
Leibniz University of Hanover
Hanover, Germany
chaaban,cms@sra.uni-hannover.de

Jörg Hähner
Institute of Organic Computing
University of Augsburg
Augsburg, Germany
joerg.haehner@informatik.uni-augsburg.de

***Abstract*—The ever increasing complexity of today's technical systems embodies a real challenge for their designers. This complexity can be regarded as the major source of unexpected system failures. Organic Computing (OC) is concerned with this complexity aiming to build robust, flexible and adaptive systems. In previous papers, we proposed a hybrid system for coordinating semi-autonomous agents using the Organic Computing concept. In this paper, we extend our prototype implementation with the aim of making it capable of handling deviations from planned (desired) behaviour. Therefore, we introduce different types of deviation that can arise. Deviations should be detected as soon as possible after their occurrence so that the controller can re-plan accordingly. In this way, the hybrid central/self-organising concept tolerates that some agents behave in fully autonomous way in the central architecture. Here, the autonomy of the agents is recognised as a deviation from the plan of the central algorithm, if the agents are not respecting this plan. Consequently, the system performance remains robust despite the occurrence of deviations.**

Keywords-Organic Computing; Hybrid Coordination; Robustness; Multi-Agent Systems

I. INTRODUCTION

The Organic Computing initiative introduces an OC system as follows [3]: "a technical system which adapts dynamically to the current conditions of its environment. It is self-organizing, self-optimizing, self-configuring, self-healing, self-protecting, self-describing, self-explaining and context-aware". Therefore, the goal of this initiative is to develop systems that are robust, flexible and adaptive at the same time utilising advantage of the organic properties of OC. In other words, OC has the objective to use principles that are detected in natural systems. In this case, nature can be considered as a model aiming to cope with the increasing complexity of the recent technical systems [3].

Organic systems use the "controlled self-organisation" design paradigm, in which the unwanted behaviour should try to be prevented, whereas the desired behaviour should be rewarded. In this regard, the robustness of OC systems is a key property, because the environments of such systems are dynamic.

Since OC systems are self-organising systems that exhibit some degrees of autonomy, the behaviour of these systems should be observed in order to take an appropriate intervention timely if necessary. The different degrees of autonomy are necessary for OC systems, so that they can adapt their behaviour to new environmental situations, where environments of such complex systems change dynamically.

This autonomy as well as disturbances, deviations in the system behaviour from that expected, and other reasons may cause an unwanted emergent behaviour [4] or the whole system may fail unexpectedly. Therefore, the system should be observed (e.g., by an observer) and controlled (e.g., by a controller), so that this emergent behaviour or the complete system failure can be prevented. Consequently, the system performance remains effective and will not deteriorate significantly or at least the system will not fail.

For this purpose, OC uses an observer/controller (o/c) architecture as an example in system design. Using the (o/c) design pattern proposed in [5], the behaviour of OC systems can be observed and controlled. A generic o/c architecture was presented in [6] to establish the controlled self-organisation in technical systems. This architecture is able to be applied to various application scenarios.

In previous papers, we introduced a system for coordinating vehicles at a traffic intersection using an o/c architecture [1][2]. The traffic intersection is regulated by a controller, instead of having physical traffic lights. Figure 1 shows a screenshot from our project.

In both earlier papers, we implemented the generic o/c architecture adapted to our traffic scenario and accomplished our experiments assuming that no deviations from plan occur in the system. In this paper, we continue with the implementation of the case when deviations from desired or planned behaviour arise in the system to completely realise our vision. Therefore, we present different types of deviation that can occur.

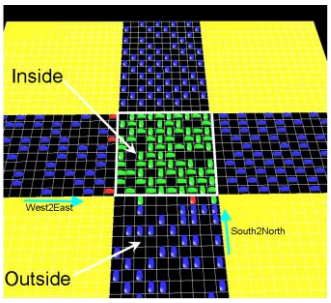

Figure 1. The traffic intersection without traffic lights

 ISBN: 978-1-61208-218-9

This paper is organised as follows. Section 2 describes our original system introduced in [1][2]. Section 3 presents a survey of related work concerning agent-based approaches used for fully autonomous vehicles within an intersection without traffic lights. Section 4 is the main part of this paper. Firstly, it describes how the observer detects deviations. The employing of neighbourhood to determine the location of deviations is depicted secondly. After that, the situation parameters including the specification of deviations and disturbances (accidents) will be explained. Accordingly, the decision making process will be presented. Section 5 introduces different metrics, which can be used to measure and test the system performance, followed by the evaluation of the system performance by means of experimental results. Section 6 draws the conclusion of this work. Finally, the future work is explicated in Section 7.

II. THE ORIGINAL SYSTEM

Previously, we proposed a new multi-agent approach which deals with the problem occurring in the system wherever multiple agents (vehicles) move in a common environment (traffic intersection without traffic lights). We presented the desired system architecture together with the technique that is to be used to cope with this problem. This architecture was an o/c architecture adapted to the scenario of traffic intersection.

The system under observation (vehicles within the centre of the intersection) is considered as a set of elements possessing certain attributes in terms of Multiagent systems. This means that every vehicle in the system is an agent. Every vehicle by itself is assumed to be egoistic (because the driver here is autonomous and he tries quickly to cross the intersection and probably he does not obey his trajectory). Therefore competition situations arise due to the egoistic behaviour (competition-based behaviour) of vehicles, which in turn leads to a traffic jam in the centre of the intersection.

A. Path Planning

Our work introduced in [1] solves a coordination problem by a central algorithm (a central-planning algorithm), using an adapted A*- algorithm that was used for path planning. Here, the path planning is considered as a resource allocation problem (resource sharing problem) where multiple agents (vehicles) move in a shared environment (traffic intersection) and need to avoid collisions.

Path planning delivers collision-free trajectories for all vehicles. Path planning has to be done only for vehicles inside the centre of the intersection. A vehicle outside the centre of the intersection has only local rules, through which this vehicle tries to move forward avoiding collisions with other vehicles.

B. Observation

The observer concentrates only on the intersection. Therefore, other observers in order to observe the agents (vehicles) on the way are not considered. In the centre of the intersection every vehicle has to obey its planned trajectory. Since deviations from the planned trajectories are possible, the monitoring is done in order to detect the deviations and to intervene dynamically through re-plan trajectories of the affected vehicles. The observer of the intersection aggregates its observations as a vector of situation parameters. These parameters are then sent to the controller.

C. Controlling

Our work presented in [2] introduced the control process of our system. The decision maker is the central part of the controller. The controller uses the decision maker to take a decision how it can intervene most suitable when it is necessary, so that the system can be influenced with respect to the given goal by the user.

However, it is worth recalling that our implementation in [1][2] assumed that all agents (vehicles) obey their planned trajectories, and consequently no deviations from plan will arise. Conversely, this paper deals with deviations aiming to complete our ambitious target (building a hybrid robust multi-agent system).

III. STATE OF THE ART

In the literature, there are enormous works concerning safety properties of usual traffic intersections that concerns only human-operated vehicles. Additionally, there are some works in connection with safety measures of autonomous vehicles within an intersection. In this paper, we focus the discussion of related work on agent-based approaches used for fully autonomous vehicles within an intersection without traffic lights.

In this regard, according to our knowledge, there are no projects that focus on the robustness of autonomous vehicles within an intersection without traffic lights, where deviations from desired (planned) behaviour occur.

A study of the impact of a multi-agent intersection control protocol for fully autonomous vehicles on driver safety is presented in [7]. In this study, the simulations deal only with collisions in intersections of autonomous vehicles. This means that the study deals only with the problem after an accident has already happened aiming to minimise the losses and to mitigate catastrophic events. It assumes that the colliding vehicle sends a signal and the intersection manager becomes aware of the situation immediately. However, it can be noted that the study has not considered the robustness of the intersection system.

Other related work to cope with the coordination problem of autonomous cars at intersection was introduced in [8]. The work proposed a priority-based algorithm that produces a collaborative behaviour between cars of an intersection without traffic lights. In this context, priorities for cars will be allocated according to the waiting times of these cars in the intersection so that car A will take a higher priority than car B if car A has a higher waiting time than car B [8].

To the best of our knowledge, this paper represents the first study towards fault-tolerant (deviation-tolerant) robust hybrid central/self-organising multi-agent systems in intersections without traffic lights using the organic computing (OC) concept.

The work in [7] deals only with collisions, whereas in this paper, the observer observes the autonomous vehicles

 ISBN: 978-1-61208-218-9

within an intersection without traffic lights in order to detect deviations. Additionally, the autonomy in this paper is recognised as a deviation, if the agents are not respecting the central plan. Consequently, the controller intervenes when it is necessary, so that the system remains demonstrating robustness.

IV. THE APPROACH

This section describes the realisation of the observation step of this paper. It presents the detection of deviations, the employing of neighbourhood to determine the location of deviations, the situation parameters which will be collected including the specification of deviations and disturbances (accidents) occurred in the system under observation, and the decision making step.

Robust systems should be fault-tolerant in order to deal with faults, deviations or disturbances and to continue working effectively and fulfilling their major tasks. In the context of this paper, fault tolerance avoids system failures in the presence of deviations from plan that occur in the system allowing the agents (vehicles) of the system to move reliably in their environment (traffic intersection without traffic lights).

The main goal of our project is keeping a multi-agent system robust when disturbances and deviations occur in the system behaviour. Agents (vehicles) have to be observed within the shared environment (intersection) through an observer (the observer of the o/c architecture), because the agents are autonomous (decentral) and they are allowed to behave in a completely autonomous way, therefore deviations from the planned trajectories (central plan) are possible.

This concept introduces a robust hybrid central/self-organising multi-agent system (hybrid coordination) solving the conflict between a central planning algorithm and the autonomy of the agents (decentral, self-organised). Here, the autonomy of the agents is recognised as a deviation from the plan of the central algorithm, if the agents are not respecting this plan (the plan is given to agents as a recommendation).

A. Detection of deviations

The observer concentrates only on the agents (vehicles) within the shared environment (centre of the intersection).

Figure 2 shows how such deviations are detected through the observer (through the detector of deviation and detector of collision) in the system.

The observer reads the planned trajectory of an agent from the trajectories-memory (should-be state), only when this agent is located within the shared environment (within the intersection). At the same time, it reads also the current travelled trajectory (actual state) of this agent, including (x_i, y_i, t_i), where the agent is at location (x_i, y_i) at time t_i. The detector of deviation in the observer compares the two states (should-be, actual) of every agent in order to detect whether any deviation from the plan occurred as in the next equation:

(should-be state) XOR (actual state)

When any deviation from the plan occurs, the detector of collision performs the next process. This process detects whether the deviation led to a collision and finds the

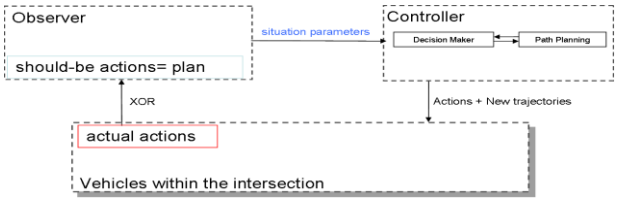

Figure 2. Detection of deviation

deviation class. The possible classes of deviation, which could be detected through the observer in this system, are: autonomy (a deviation from plan), accident (a disturbance) or autonomy with accident (a deviation with a disturbance).

Afterwards, the observer aggregates its observations as a vector of situation parameters describing the actual states and sends it to the controller which selects the more suitable actions and sends it to the system with the new trajectories.

It is important to mention that the controller will decide whether the detected deviations can be tolerated with respect to the safety distance around the agents (vehicles) as described in section (Decision Making).

The deviation detector uses the idea of neighbourhood in order to compare the two states (should-be and actual states) of every agent as described in the next section.

B. Neighbourhood

The term neighbourhood is used in this paper. It is used wherever multiple agents (e.g., robots, vehicles, etc.) move in a common environment (intersection). This paper reaps the benefit of employing "neighbourhood" so that the places of occurred deviations can be determined (the direct neighbourhood, or the second neighbourhood, etc).

Here, the neighbourhood is a square-shaped neighbourhood which can be used to define a set of cells (C) surrounding a given cell c_0 with (x_0, y_0) coordinates; whereas the common environment (the intersection) is a square grid.

When an agent A (vehicle V) is located in a cell c_0 (x_0, y_0) of the intersection, then the neighbourhood Nc_0 of this cell is the set of cells $C = \{c_i (x_i, y_i)\}$ that can be seen by this agent (vehicle) from the central cell c_0 (x_0, y_0). Consequently, the neighbours of this agent A (vehicle V) is a set of agents A_i (vehicle V_i) which are located in this neighbourhood Nc_0.

The neighbourhood has a distance (radius) which determines to which extent is this neighbourhood limited. This extent represents the view of an agent (vehicle). In a metric space of cells M = (C, d), a set Nr (c_0) is a neighbourhood of a cell c_0 if there exists another set of cells with centre c_0 and radius r, so that

$$N_r(c_0) = N(c_0; r) = \{c \in C \mid d(c, c_0) < r\}$$

Figure 3 shows the idea of neighbourhood which is used in this paper. This definition of neighbourhood can be seen as the well-known Moore neighbourhood (also known as the 8-neighborus) where the distance (radius) is 1. In reference [9], the Moore neighbourhood of range r is defined by:

$$N^M_{(x_0, y_0)} = \{(x, y) : | x - x_0 | \leq r, | y - y_0 | \leq r\}$$

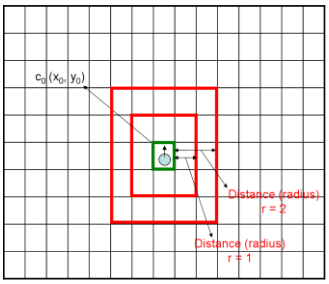

Figure 3. The used neighbourhood

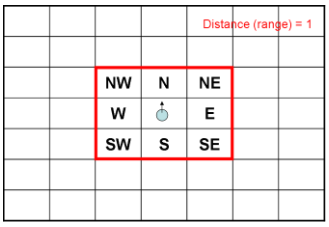

Figure 4. The direct (first) neighbourhood

Here, the neighbourhood is a set of cells surrounding a given cell (x_0, y_0). Moore neighbourhoods for ranges r = 0, 1, and 2 are illustrated in Figure 3. It can be inferred that the number of cells in the Moore neighbourhood of range r is the odd squares $(2r + 1)^2$.

It can be seen that the number of cells in the Moore neighbourhood of range r=0 is 1, i.e., it contains only the cell c_0 (x_0, y_0) where the agent (vehicle), whose neighbourhood is under search, is located. However, it contains 9 cells in the Moore neighbourhood of range r=1, i.e., it contains 8 neighbours and the cell c_0 (x_0, y_0) itself. In this case of Moore neighbourhood (r=1), the neighbourhood can be called the first or direct neighbourhood. The direct neighbourhood is used as depicted in Figure 4.

Here, the agent (vehicle) is located in the central cell and the 8 neighbours of this agent are named according to the direction in which there are the following neighbours:

N: North, E: East, S: South, W: West.

NW: North West, NE: North East, SE: South East, SW: South West.

Therefore, the neighbourhood of the central cell c_0, where the agent (vehicle) is located, can be defined as follows:

$$N_1(c_0) = N(c_0; 1) = \{c \in C \mid d(c, c_0) < 1\}$$

$$N_1(c_0) = \{ c_0, N, E, S, W, NW, NE, SE, SW\}$$

By the range r=2, the Moore neighbourhood contains 24 neighbours and the cell c_0 (x_0, y_0) itself (the second neighbourhood).

When the observer detects that an agent (vehicle) is located (the actual state) in a cell (e.g., the cell N), which belongs to the direct neighbourhood $N_1(c_0)$, instead of in their planned cell (e.g., the central cell c_0 that is the should-be state), then it sends to the controller that the deviation of the agent (vehicle) is within the first (direct) neighbourhood $N_1(c_0)$. In a similar way, the observer sends to the controller that the deviation of the agent (vehicle) is within the second neighbourhood, if it detects that the agent (vehicle) is located in a cell which belongs to the second neighbourhood $N_2(c_0)$ instead of in their planned cell c_0.

C. Situation Parameters

The situation parameters contain the class of the detected deviation and the (x, y, t) of the deviation (the new state), if the observer has detected a deviation. When the controller gets the situation parameters containing a deviation message, then it activates the decision maker and plans new trajectories, if needed, and sends to the system the more suitable actions with the new trajectories.

The situation parameters represent the global description of the current situation of the system under observation and include five parameters:

[Deviations, Accidents, Exceptions, predictions, confidence interval]

- Specification of the detected deviations (unplanned autonomous behaviour).
- Specification of the detected disturbances (accidents).
- Exceptions: e.g., an emergency car.
- Predictions: e.g., the arrival time of the emergency car to the intersection (that is future consequences).
- Confidence interval: e.g., currently normal planning but after two minutes, a special plan for the emergency car shall be activated (that is future consequences).

D. Deviation specification

In this paper, we focus only on deviations from planned behaviour (disturbances, accidents, will be considered in future work). The specification of the deviations includes the following features:

- If any deviation from plan was detected: {true, false}.
- If a deviation was occurred (true), then the begin time of the deviation occurrence, Start (t) and the end time if it's over, End (t).
- The location of the deviation where it occurred, the coordinates (x, y).
- The type of the deviation (deviation from the planned trajectories).
- The result of the deviation. If the deviation caused an accident or not: {Accd, No. Accd}.

In this regard, there are four possible deviation types. According to the time of the deviation occurrence, it can recognise two deviation types. First, vehicles can change its speed (change speed) trying for example to cross the intersection more quickly than planned. Second, vehicles can stop (making stop) trying to avoid a potential collision with another vehicles in the intersection. However, according to the location of the deviation, two other deviation types can be recognised. First, vehicles can change its lane (change lane) trying for example to leave a full lane of vehicles in order to drive quickly as long as possible. Second, vehicles can change its direction (change direction) trying for example to avoid a potential collision with another vehicles in the intersection or trying to avoid a traffic jam in the intersection.

 ISBN: 978-1-61208-218-9

E. Decision Making

The controller uses the decision maker to take a decision how it can intervene most suitable when it is necessary, so that the system can be influenced with respect to the given goal by the user. The given goal of the user in the introduced application scenario (the intersection) is to keep the system demonstrating robustness in spite of fully autonomous behaviour (causes deviations from plan) and disturbances (accidents) which could appear in the intersection system. In addition, it aims to get autonomous traffic as possible with low delays.

The decision maker is activated when the controller gets the situation parameters from the observer containing a deviation message. On the other side, when there is no deviation, this means that everything is as planned and the decision maker will not be used in this case.

The controller has to intervene on time if it is necessary (decision maker unit) and to select the best corrective action (it makes a decision whether a replanning is required and it uses also the path planning unit if needed) that corresponds to the current situation so that the system performance remains acceptable and the target performance of the system is maintained. Here, the controller has the capability of fault-tolerance (deviation-tolerance) and consequently it decides whether the detected deviations can be tolerated with respect to the free positions (safety distance) around the agents (vehicles). It tolerates a deviation unless the limit of the safety distance is not exceeded through the deviated agent (vehicle). The controller sends to the system the appropriate actions with the new planned trajectories according to the actions table.

V. PERFORMANCE EVALUATION

In this section, we present an initial evaluation of our system using the model of a traffic intersection, which was designed and described in our earlier paper [1]. We include only our main result aiming to deal with deviations from planned (desired) behaviour of agents (vehicles). In future work, we intend to present a more complete empirical evaluation including experiments for measuring the robustness of the system, in which deviations from plan occur and disturbances (accidents) appear in the intersection system.

A. System Performance Metrics

This section will prove the performance of the intersection system presenting an empirical evaluation including experiments with three metrics: throughput, main waiting time and main response time. Here, the first metric, the throughput, is required for estimating the overall reduction of the system's performance, in which deviations from the plan of the controller occur. According to the intersection system, throughput is the total amount of vehicles that left the intersection (simulation area) over time, whereas the mean waiting time is the mean waiting time (ticks or iterations) needed by vehicles to traverse the intersection. The response time has two parts. Firstly, the path planning time, it is the time to search for trajectories, i.e., the time between the moment when the path planning unit in the controller of the o/c architecture gets messages (requests) from the system (vehicles) and the moment when it sends appropriate trajectories to the system (vehicles). Secondly, the observation/controlling time, it is the needed time by the observer to detect a deviation and the needed time by the controller to re-plan the affected trajectories.

The measurement of the three metrics will be made using several values of the simulation parameter, the maximum number of vehicles, employing the evaluation scenario I (Equal-Equal) described in [1]. Accordingly, the traffic flow rates (traffic levels) of vehicles in south-north and west-east directions is equal, namely 5 vehicles/tick. However, the measurement has been repeated in the cases that the maximum number of vehicles in each direction is equal, namely 20, 40, 80, and 100 vehicles (40, 80, 160, 200 vehicles in both directions). The three used metrics have been measured after 3000 ticks.

B. Test environment

As a test environment, a Pentium 4 personal computer with 2.8 GHz speed and 2 GB RAM has been used to perform the simulation of the traffic application scenario.

C. Test situation

In order to deal with deviations from plan, we assume that vehicles violate (do not obey) their planned trajectories in the centre of the intersection but there are no accidents in the intersection.

In this paper, the desired type of potential deviations in the simulation of the traffic system is (change speed). That means, vehicles can change its speed trying for example to cross the intersection more quickly than planned. With respect to this type of deviations, vehicles that can only make one move (its planned speed is one move) in one tick (a single time step) are trying to make two moves in the same length of simulation time (one tick). For example, a vehicle moves two steps forward, if there is no vehicle in the next two cells in front of it in the intersection. Otherwise, it moves only one step (as its planned speed) if no vehicle only in the next cell in front of it. However, deviations do not cause any accident.

Then, we will measure the system performance and compare the two cases, the system performance with and without deviations. This comparison was managed between the measured values of the first case (without deviations) and the measured values of the second case (with deviations) using the three metrics mentioned above: throughput, mean waiting time and mean response time.

Such comparison can lead to discover the effect of non-compliance with the central plan (planned trajectories of vehicles). Consequently, the comparison here is between the first case (without deviations), which is full central planning and the second case (with deviations), which is hybrid coordination (central and decentral). Therefore, this comparison will be used to determine whether the system performance remains acceptable (will not deteriorate considerably) despite the occurrence of deviations.

 ISBN: 978-1-61208-218-9

# Vehicles (Max. Number of Vehicles)	Si1 Without Deviations (vehicles)	Si2 With Deviations (vehicles)
40	989	989
80	1950	1951
160	3849	3864
200	4800	4786

TABLE I. THE COMPARISON OF THE SYSTEM THROUGHPUT

D. Results of throughput measurement

Table I shows the comparison of the system throughput measured after 3000 ticks between the first case Si1 (without deviations) and the second case Si2 (with deviations) with varying amounts of the maximum number of vehicles in both directions from 40 to 200 vehicles.

In this regard, the same behaviour of the system throughput applies to both cases Si1 (without deviations) and Si2 (with deviations) can be seen. Consequently, the values are roughly identical in both cases. That means, the system throughput by the second case Si2 increases almost always linearly with the number of vehicles despite deviations from the planned trajectories (due to the autonomous vehicles).

This emphasises that no degradation of the system throughput was established when vehicles make deviations (drive more speedily than the plan) and thus do not obey their planned trajectories. Therefore, it is inferred that the central plan (the path planning by means of a central planning algorithm) was optimal. However, no improvement of the system throughput was found. The reason for this is that the vehicles which drive more speedily than planned block other vehicles in the neighbourhood to obey their planned trajectories. Therefore, increase the speed of a vehicle will be at the expense of the speed of other vehicles in the next neighbourhood and thus it may lead to delays of these vehicles.

E. Results of mean waiting time measurement

Table II shows the comparison of the mean waiting times and the standard deviations of all vehicles, that left the intersection measured after 3000 ticks between the both cases Si1 and Si2 with varying amounts of the maximum number of vehicles in both directions from 40 to 200 vehicles. Additionally, Figure 5 shows the same comparison of the mean waiting time as diagram.

Here, it can be seen the same behaviour of the waiting times applies to both cases Si1 and Si2. Consequently, the mean waiting times are roughly identical by both cases. More accurately, there is very small increase by Si2 due to deviations. That means, the mean waiting times by the second case Si2 increase but very slightly despite the deviations from the planned trajectories (due to the autonomous vehicles).

The very small increase of the mean waiting times by Si2 can be traced back to the deviations of vehicles (drive more speedily than planned) which lead in turn to block other vehicles in the neighbourhood causing longer delays than those intended (planned). This confirms the conclusion that the central plan (the path planning for the vehicles) was optimal.

	Si1 Without Deviations		Si2 With Deviations	
# Vehicles	Mean Waiting Time (Ticks)	Std. Deviation	Mean Waiting Time (Ticks)	Std. Deviation
40	0.59	0.80	0.59	0.80
80	0.84	0.53	0.89	1.09
160	1.13	1.16	1.21	1.23
200	1.43	1.39	1.50	1.32

TABLE II. THE COMPARISON OF THE MEAN WAITING TIMES AND THE STANDARD DEVIATION OF ALL VEHICLES THAT LEFT THE INTERSECTION

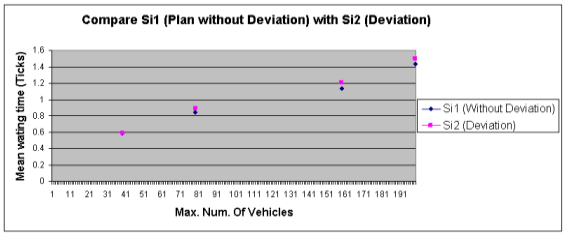

Figure 5. The comparison of the mean waiting times and the standard deviation of all vehicles that left the intersection

Here, it can be seen the same behaviour of the waiting times applies to both cases Si1 and Si2. Consequently, the mean waiting times are roughly identical by both cases. More accurately, there is very small increase by Si2 due to deviations. That means, the mean waiting times by the second case Si2 increase but very slightly despite the deviations from the planned trajectories (due to the autonomous vehicles).

The very small increase of the mean waiting times by Si2 can be traced back to the deviations of vehicles (drive more speedily than planned) which lead in turn to block other vehicles in the neighbourhood causing longer delays than those intended (planned). This confirms the conclusion that the central plan (the path planning for the vehicles) was optimal.

F. Results of mean response time measurement

Figure 6 shows the comparison of the mean response time between the both cases Si1 and Si2 with varying amounts of the maximum number of vehicles in both directions from 40 to 200 vehicles. This comparison was made by the evaluation scenario I (Equal-Equal) after 3000 ticks using the reservation way (AllTrajectoriesVector) described in [1].

Here, different behaviour of the response times between Si1 (without deviations) and Si2 (with deviations) is clearly seen. More accurately, there is an increase by Si2 due to deviations. That means, the mean response times by Si2 increase clearly but reasonably despite deviations from the planned trajectories (due to the autonomous vehicles).

The reasonable increase of the mean response times by Si2 can be attributed to the long time (comparing with the Si1 without deviations) needed to detect deviations occurred in the system and to re-plan all affected trajectories. It can be seen that by (100-100) vehicles in the intersection the mean response time is less than 15 ms, which can be considered a reasonable value for the today's modern computing devices.

 ISBN: 978-1-61208-218-9

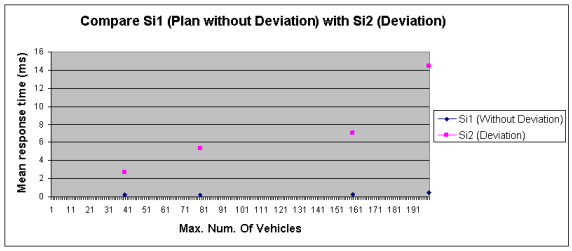

Figure 6. The comparison of the mean response time after 3000 ticks between the first and second cases

VI. CONCLUSIONS

In this work, the hybrid central/self-organising concept aims to increase the autonomy of agents in the central architecture. This means, the hybrid concept tolerates that some agents behave in fully autonomous way in the central architecture. It solves the conflict between a central planning algorithm (a component in the controller) and the autonomy of the agents (the entities of the system under observation and control).

In this paper, deviations from the plan occur but there are no accidents occur in the intersection. That means, vehicles do not obey their planned trajectories (i.e., decentral due to autonomous vehicles). Here, actual trajectories of vehicles will be observed identifying any deviations from plan in order to make replanning by means of the path planning algorithm. Here, vehicles try to change its speed crossing the intersection more quickly than planned (deviation type is change speed). Therefore, we extended our prototype implementation with the aim of making it capable of handling potential deviations.

In this regard, a comparison was made between the first case Si1 (without deviations) and the second case Si2 (with deviations). So, the comparison here was between Si1, which is fully central planning and Si2, which is hybrid coordination (central and decentral). This comparison was made using three metrics, throughput, mean waiting time and mean response time.

The present implementation shows high success potential by detecting deviations from desired (planned) behaviour and consequently to make replanning. By this comparison, the system of central planning shows approximately the same performance as the system of decentral planning when only deviations from the central plan occur but no disturbances (accidents) occur. Based on this, it can be concluded that a local problem (e.g., small deviation from plan) can be solved at local level.

VII. FUTURE WORK

In this paper, we implemented the generic o/c architecture adapted to our traffic scenario and accomplished our experiments assuming that no accidents (disturbances) occur in the system environment (intersection). Therefore, the next step is to continue with the implementation of the case when accidents arise, in addition to deviations from plan, in the intersection aiming to completely realise our vision. Then, we will measure the system performance and compare the two cases, the system performance with deviations and accidents (disturbances) on one side and the system performance without deviations or accidents from the other side. This comparison will be used to determine whether the system performance remains effective (robust) despite disturbances and deviations occurred in the system (intern) or in the environment (extern). Simultaneously, an appropriate metric for the quantitative determination of the robustness will be developed.

REFERENCES

[1] Yaser Chaaban, Jörg Hähner, and Christian Müller-Schloer. “Towards fault-tolerant robust self-organizing multi-agent systems in intersections without traffic lights”. In Cognitive09: proceedings of The First International Conference on Advanced Cognitive Technologies and Applications, November, 2009, pp. 467-475, Greece. IEEE.

[2] Yaser Chaaban, Jörg Hähner, and Christian Müller-Schloer. “Towards Robust Hybrid Central/Self-organizing Multi-agent Systems”. In ICAART2010: proceedings of the Second International Conference on Agents and Artificial Intelligence, Volume 2 , pp. 341-346, January 2010 Spain.

[3] CAS-wiki: Organic Computing. http://wiki.cas-group.net/index.php?title=Organic_Computing, [retrieved: February, 2012]

[4] Moez Mnif, Urban Richter, Jürgen Branke, Hartmut Schmeck, and Christian Müller-Schloer. “Measurement and control of self-organised behaviour in robot swarms”. In Paul Lukowicz, Lothar Thiele, and Gerhard Tröster, editors, Proceedings of the 20th International Conference on Architecture of Computing Systems (ARCS 2007), volume 4415 of LNCS, pp. 209–223. Springer, 2007.

[5] Christian Müller-Schloer. “Organic computing: on the feasibility of controlled emergence”. In CODES+ISSS ’04: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp. 2–5. ACM, 2004.

[6] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. “Towards a generic observer/controller architecture for organic computing”. In Christian Hochberger and Rüdiger Liskowsky, editors, INFORMATIK 2006 - Informatik für Menschen!, volume P-93 of GI-Edition - Lecture Notes in Informatics (LNI), pp. 112–119. Bonner Köllen Verlag, 2006.

[7] Kurt Dresner and Peter Stone. “Mitigating catastrophic failure at intersections of autonomous vehicles”. In AAMAS ’08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pp. 1393–1396, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[8] Emre Cakar, Jörg Hähner, and Christian Müller-Schloer. “Creating collaboration patterns in multi-agent systems with generic observer/controller architectures”. In Autonomics ’08: Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems, pp. 1–9, Belgium, 2008.

[9] Moore Neighborhood in the WolframMathworld. http://mathworld.wolfram.com/MooreNeighborhood.html, [retrieved: March, 2012].

 ISBN: 978-1-61208-218-9