Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1471–1480 DOI: 10.15439/2016F126
ACSIS, Vol. 8. ISSN 2300-5963

Self-Organizing Redistribution of Bicycles in a Bike-Sharing System based on Decentralized Control

Thomas Preisler, Tim Dethlefs and Wolfgang Renz
Faculty of Engineering and Computer Science,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
Email: {thomas.preisler,tim.dethlefs,wolfgang.renz}@haw-hamburg.de

***Abstract*—Currently, bike-sharing systems undergo a rapid expansion due to technical improvements in the operation combined with an increased environmental and health awareness of people. When it comes to the acceptance of such systems the reliability is of great importance. It depends heavily on the availability of bicycles at the stations. But, in spite of truck-based redistribution efforts by the operators, stations still tend to become full or empty, especially in rush-hour situations. This paper builds upon an incentive scheme that encourages users to approach nearby stations for renting and returning bikes, thereby redistributing them in a self-organized fashion. A cooperativeness parameter is determined by the fraction of users that respond to an incentive by choosing the proposed stations. It uses a decentralized control process to calculate alternative rent and return stations for each of the stations. These alternatives are then proposed to the users when they approach an empty or full station. The approach is based on a decentralized control framework that allows to equipping different distributed software systems with the control capabilities needed to realize the coordination efforts required to achieve the desired self-organizing properties.**

I. Introduction

RECENT challenges like climate changes, declining supplies of fossil fuels, noise emissions and congestion lead to discussions about individual means of transportation in urban areas. Especially bicycles (bikes in the following) have received an increased attention in city transportation, as they offer a healthy and environment-friendly way of transportation and allow to reach areas in cities that do not have direct access to public transportation. Combined with technical improvements of the underlying information systems, this results in a rapid extension of bike-sharing systems worldwide [1]. Obviously, bikes have drawbacks in comparison to other modes of transportation, as the usage of bikes strongly depends on weather conditions and the topography of the targeted area. This makes bikes more suitable for short trips [2]. As mentioned before, the increasing success of bike-sharing systems depends strongly on the introduction of information systems supporting the whole renting process (finding available bikes in the departure area as well as renting and returning them) [3]. Today, many cities aim at implementing bike-sharing systems in order to improve inner-city air quality and to reduce congestion [4].

The main challenge for the operation of modern bike-sharing systems in big cities is to ensure the *availability* of bikes at the stations. In rush-hour situations, stations may run out of bikes while others become full, thus reducing the overall *reliability* of the systems. Therefore, the planning and operation of redistribution attempts is essential to ensure reliability and user satisfaction. There are several attempts that have been tested to overcome these problems in scientific research as well as in practice [1] (cf. Section II).

This paper extends previous work [5], where an incentive scheme was investigated, that encourages users to approach nearby stations for renting and returning bikes, thus redistributing them in a self-organizing way. This work is extended by two aspects: First, a decentralized control framework is introduced that allows the declarative description of decentralized coordination processes to control the required coordination efforts among the participating entities in order to achieve the desired self-organizing behavior. Thereby, it shall support different types of heterogeneous applications and systems. The framework is used to replace the coordination processes that were tailored especially for the used RinSim simulator [6] presented in [5] by declarative, generic ones. Second, based on the redesigned control processes, the efficiency of the self-organizing redistribution approach depending on a circular communication range coordination parameter is examined. The communication range determines which other bike stations are within reach of a certain bike station and therefore, receive the status updates emitted from this station as part of the coordination process. There is a direct 1:1 mapping between the communication range and the maximum distance a user is detoured when an alternative rent or return station is proposed to him. Thus, minimizing the communication range is of concern when to ensure user cooperation and satisfaction.

Following the approach described in [5] a microscopic simulation of an idealized Monday based on data from Washington, D.C.'s bike-sharing system (2014) is realized. The simulation will be used to describe the application of the decentralized control framework and to measure the impact of the communication range as a coordination parameter. Washington, D.C.'s bike-sharing system was chosen as a base for the simulation as all data concerning the system is freely accessible over the *Capital Bikeshare Dashboard* [7]. The *Capital Bikeshare* system has been started in September 2010 and until May 2013 it was the largest bike-sharing service offered throughout the United States [8]. In 2014, the system had 345 stations and about between 2400 and 2900 bikes were available for usage. Like many other bike-sharing system the pricing is based on the principle that the first 30 minutes of a rental are for free (except a fixed membership fee). Each additional 30 minutes require an extra fee. To ensure the reliability of the system and therefore, both the availability of bikes and free docks at the stations, Capital Bikeshare uses trucks to redistribute the bikes [9]. Fig. 1 shows the rebalancing efforts ventured by Capital Bikeshare in 2014. The figure shows that the operation of such a bike-sharing systems requires a significant amount of redistribution efforts.

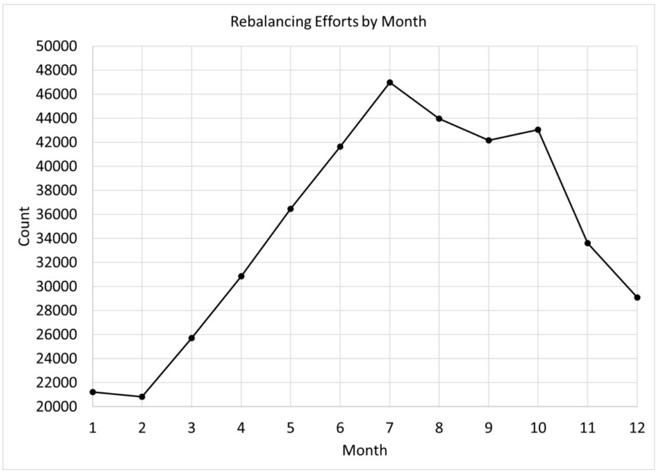


Fig. 1. Capital Bikeshare rebalancing efforts in 2014 (data taken from [7]).

However as shown in Fig. 2 stations still tend to become empty or full. Here the number of full and empty instances per month in 2014 are shown. From the data it becomes visibly, that the amount of empty stations is higher than the amount of full stations, indicating that the stations may be designed to have spare capacities to increase the chance that a bike can be returned at a station. In conclusion, the figure shows that, despite the redistribution efforts that are already carried out, there is potential for further improvement. This can either be an increasing number of truck-based redistributions or the introduction of new approaches, especially to overcome the problem of empty stations, e.g., by a self-organizing approach for the redistribution of bikes by the users.

The remainder of this paper is structured as follows: the next section will introduce related work in terms of the operation and planning of redistribution attempts as well as approaches dealing with the realization of self-organizing behavior based on decentralized control. Section III presents a decentralized control framework designed to allow the construction of decentralized coordination processes for different types of applications and systems. In Section IV, a Multi-Agent-based

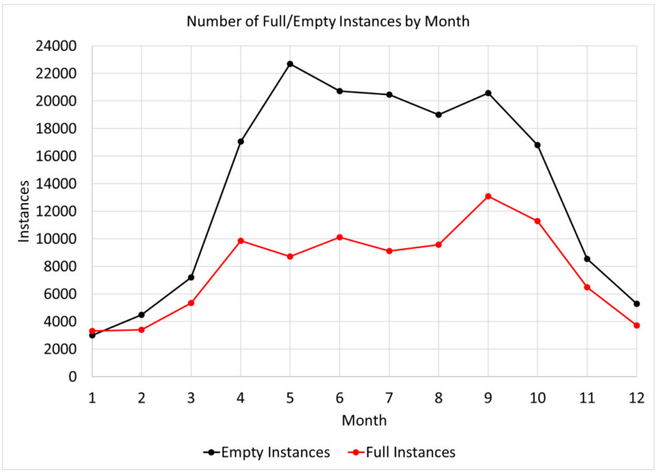


Fig. 2. Number of full/empty instances in 2014 (data taken from [7]).

simulation system of Washington D.C.'s bike-sharing system is introduced and it is described how the decentralized control framework is used to realize the self-organizing redistribution of bikes in the simulation system. Section V presents the results of the simulation and evaluates the impact of the communication range coordination parameter to the self-organizing strategy. Finally, Section VI concludes the paper and presents an outlook on future work.

II. Related Work

Several attempts have been ventured out to overcome the previous mentioned problem of maintaining reliability in bike-sharing systems. The authors of [10] used clustering techniques to identify shared behaviors across stations in order to predict short-term station usage for Barcelona's *Bicing* system. Thereby, they provide a spatiotemporal analysis of 13 weeks of bike station usage to sense and predict the rush on certain stations depending on their location and the time of day. Thus, supporting the operation of the system by providing planning support for the distribution of bikes. Similar work, also focusing on Barcelona's Bicing system was done in [11]. The authors analyzed human mobility data in an urban area using the amount of available bikes at the stations. Based on the data sampled by the system operator's website temporal and geographic mobility patterns within the city have been detected. These patterns were applied to predict the number of available bikes at the stations ahead. The timeliness of the whole bike-sharing topic is elaborated in [12] where the characteristics and commonalities between particular bike-sharing systems with a view to deriving influences on the sustainability of systems is explored. The empirical study analyzes bike-sharing systems in five Chinese cities. As China is suffering from severe negative consequence of high private vehicle usage in large and densely populated cities, it would greatly benefit from an environmentally-friendly way of transportation like bicycling. China has a long history of bike usage in the country and therefore, it provides a great potential for such a green form of travel to be part of public and private transportation. Therefore, the authors of [12] analyze the effect of different bike-sharing systems in China and draw conclusions based on their success for the development of new systems.

In terms of building self-adaptive and self-organizing systems, there are several approaches which deal with the associated challenges. Research areas like Autonomic [13] or Organic Computing [14] provide approaches to solve these challenges. Both approaches rely on different types of feedback loops based on (usually) centralized control elements. According to [15] feedback loops are a key design element within a distributed system in order to exhibit adaptivity. Feedback loops normally consist of three main components: (1) Sensors are in charge of observing the behavior and the (current) status of the component, respectively the environment it is situated in. (2) Actuators can change the configuration of the system, which can lead to changes in the component's behavior. (3) A computing entity serves as a connector between the system input (sensor) and the output (actuator). It can be very different with regards to its internal architecture and abilities (cf. [16]). The importance of decentralized control to achieve requirements like resilience, robustness and scalability in large distributed systems has been identified in [17]. The work presented there distinguishes decentralized self-adaptive solutions from their centralized counterparts and also proofs some of the key research challenges for the realization of decentralized self-adaptation.

Related work in terms of decentralized coordination is among others presented in [18]. There a framework for the decentralized coordination of ubiquitous web services is proposed. It is based on an Event-Condition-Action (ECA) approach and relies on an XML-based language for describing ECA rules that are embedded in web service-enabled devices. Another example for a self-organizing infrastructure that offers coordination capabilities, inspired by chemical reactions is the *TuCSoN* coordination space concept [19]. It relies on a multiplicity of independent communication abstractions, called tuple centers. These can be spread over Internet nodes and are used by agents to interact with each other. TuCSoN exploits tuple centers as its coordination media, where a tuple center enhances a tuple space with a behavior specification. Therefore, the tuple centers are a communication abstraction whose behavior can be defined to embed an overall law of coordination. This is similar to the approach presented in this paper which utilizes coordination media as communication abstractions. Also similar is the propagation of a clean separation of concerns between application and coordination logic as introduced by [20]. The authors of [20] propagate a loose coupling between the core functionality of an application (computation) and the coordination. Thereby coordination is an orthogonal aspect w.r.t. to the computation when it comes to the realization of distributed systems. According to [20] this increases the generality when the coordination is swapped in a separate model.

However, there is a lack of approaches that support decentralized coordination in general regardless of the used technology and design patterns. The authors of [21] for example present a decentralized framework for the dynamic composition and coordination of autonomic agent applications. As a first step toward such a general decentralized control framework, that will be proposed in the next section, previous work dealt with a middleware supporting the construction of decentralized control in self-organizing system based on the concept of Active Components [22]. Active Components combine the autonomous behavior known from software agents with the service provider paradigm from the Service Component Architecture (SCA). A more detailed description about the concept of Active Components is given in [23].

III. DeCoF: A decentralized Coordination Framework

Today's distributed systems are characterized by an increasing size and complexity, which requires novel engineering approaches. The utilization of self-organizing processes has been proposed to enable adaptiveness of inherently decentralized systems [24]. Self-organization refers to physical, biological and social phenomena, where global structures arise from local interactions of autonomous individuals [25]. It has turned out to be a promising paradigm for the development of advanced distributed applications and systems with strongly decentralized control and high demands for self-adaptive behavior.

The designed decentralized coordination framework is based on the concept that the self-organizing dynamic that causes a system to adapt to external and internal influences is controlled by decentralized coordination processes. The processes describe the self-organizing behavior that continuously structures, adapts and regulates aspects of the application. They instruct a set of decentralized Coordination Media and Coordination Endpoints. Coordination Media deal with the interactions between the components (information propagation), while the Coordination Endpoints handle the adaptation of the components (local entity adaptation). Together they control the microscopic activities of the components, which on a macroscopic level lead to the manifestation of the intended self-organizing dynamic. The integration of the Coordination Endpoints and Media is prescribed by declarative defined coordination processes which structure and instruct their operations (cf. Section IV-A for an example of such a declarative coordination process description).

The Decentralized Coordination Framework (DeCoF) emerges from a tailored programming model for the software-technical utilization of coordination processes as reusable design elements in Multi-Agent Systems (MAS). The DeCoMAS (Decentralized Coordination for Multi-Agent Systems) [26] architecture introduces concepts like Coordination Media for the propagation of Coordination Information and Coordination Endpoints for the observation and adaptation of the local entities. But while the DeCoMAS architecture is especially designed to equip BDI[1]-agent system with coordination processes and therefore, is limited to such systems, DeCoF aims at supporting distributed systems in general. Thereby, different and also heterogeneous software components in general are supported, allowing to equip not only MAS but component based systems in general with decentralized coordination processes to extend them with self-organizing capabilities. The current reference implementation supports software components written in *Java*. But, it is also applicable to other programming languages in general.

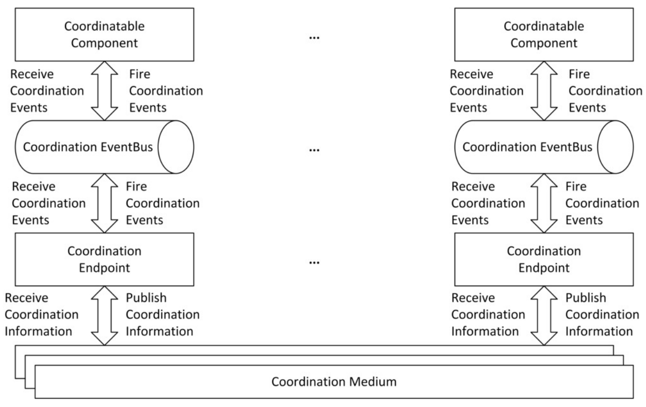


Fig. 3. Architecture of the Decentralized Coordination Framework

Fig. 3 shows the conceptual architecture of the proposed framework. Components respectively agents that should be equipped with coordination capabilities to realize self-organizing behavior based on the aforementioned concepts are labeled as *Coordinatable Components*. As the framework aims at supporting various types of MAS resp. distributed systems in general, there are no inherent characteristics that could be used to monitor or control the behavior of the agents respectively components, e.g. different types of MAS use different scheduling and life-cycle mechanisms, while component-based systems might lack them at all. Therefore, the concept of *Coordination Events* is introduced. These are events that are fired by a coordinatable component, whenever something relevant for the coordination happened inside the component. A coordination event (ce) is a tuple with the length 2 (double or 2-tuple) containing contextual data about the specific coordination event (cd) as well as a representation of the event's originator (eo). A coordination event is thus defined as: $ce = (cd, eo)$.

Following a separation of concerns between the application and the coordination logic as propagated by [20], the actual processing of the coordination events is handled by a related *Coordination Endpoint*. The coordination endpoints are loosely coupled to the coordinatable component via a so called *Coordination Event Bus*. An event bus[2] allows publish-subscribe-style communication between components without requiring the components to explicitly register with one another (and thus be aware of each others). The separation of concerns requirement is fulfilled by the loosely coupling between the coordinatable component and the coordination endpoint realized by the coordination event bus. Thus, the component is only responsible for realizing the application logic and do not need to have knowledge about (the present of) the coordination endpoint.

When a coordinatable component fires a coordination event, it is received by the related coordination endpoint via the coordination event bus. The endpoint then processes the event according to a prescribed coordination process definition. The process definitions defines how different coordination events have to handled by the endpoints. These descriptions contain instructions on *how* to distribute the coordination event to *which* other coordinatable components. The *how* is described by indicating what kind of *coordination medium* should be used for the information dissemination. As described before, coordination media deal with the information propagation among the components. The *which* is realized with a role-concept. A coordination process definition specifies various roles that components might adopt. Thereby, a component can have multiple roles and a role may be carried out by various component types. So to process a coordination event the endpoint encapsulates it and enriches it with additional information about the originating coordination endpoint. The resulting *Coordination Information* (ci) is a 2-tuple containing the coordination event (ce) and information about the originating endpoint (oe), thus is defined as: $ci = (ce, oe)$. Besides prescribing which coordination event, originating from which coordinatable components should be published to which other components, a coordination process definition also prescribes which type of coordination event should be triggered in the receiving components. How the coordination information are actually propagated is part of the implementation of the actual coordination medium. This regards the technical realization of how the information should be distributed, as well as how the subset of receivers is selected. Therefore, simple coordination medium relying on a network-topology for the information dissemination as well as complex ones, where the dissemination of the information relies on, e.g. diffusion processes in an (virtual) environment are possible.

Fig. 4 shows an UML class diagram of the relevant classes and interfaces of the framework. A component that should be equipped with coordination capabilities has to implement the `ICoordinatable` interface. It requires the component to implement two functions. The `getId` function returns an unique string that identifies the component. The `handleCoordinationEvent` function is called whenever a coordination event relevant for the component has been received by its coordination endpoint. Here the

[1]The Belief, Desire, Intention software model is developed for programming intelligent agents. It is characterized by the implementation of an agent's beliefs, desires and intentions and uses these concepts to solve a particular problem in agent programming [27].

[2]See: https://www.github.com/google/guava/wiki/EventBusExplained (accessed April 20, 2016)

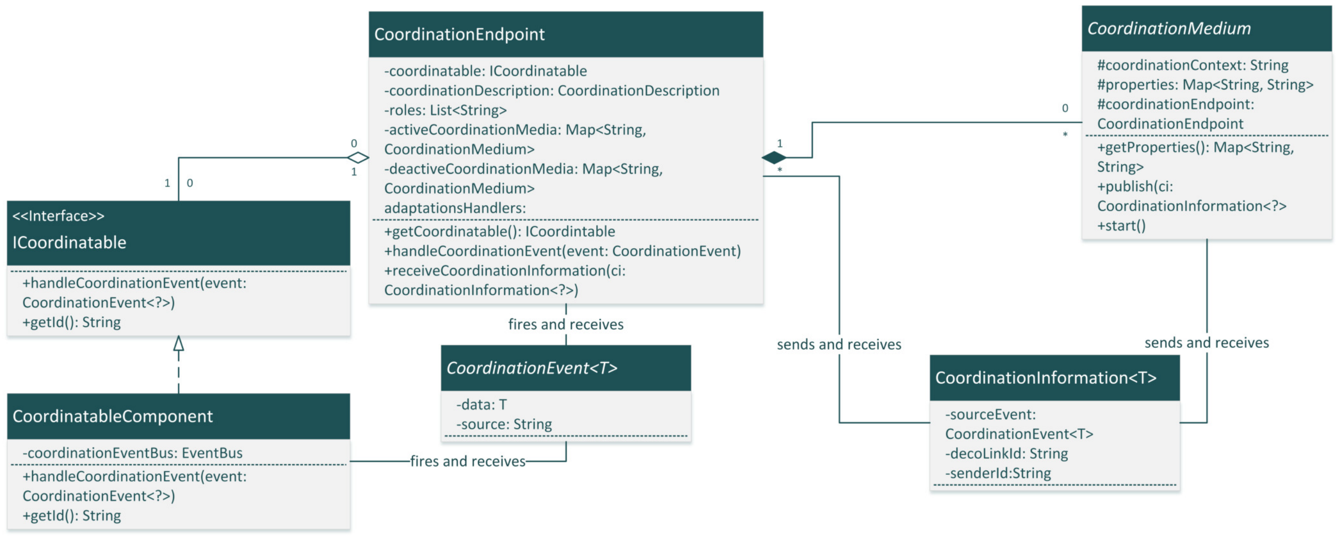

CoordinationEndpoint
-coordinatable: ICoordinatable
-coordinationDescription: CoordinationDescription
-roles: List<String>
-activeCoordinationMedia: Map<String, CoordinationMedium>
-deactiveCoordinationMedia: Map<String, CoordinationMedium>
adaptationsHandlers:
+getCoordinatable(): ICoordintable
+handleCoordinationEvent(event: CoordinationEvent)
+receiveCoordinationInformation(ci: CoordinationInformation<?>)
CoordinationMedium
#coordinationContext: String
#properties: Map<String, String>
#coordinationEndpoint: CoordinationEndpoint
+getProperties(): Map<String, String>
+publish(ci: CoordinationInformation<?>
+start()
<<Interface>>
ICoordinatable
+handleCoordinationEvent(event: CoordinationEvent<?>)
+getId(): String
CoordinatableComponent
-coordinationEventBus: EventBus
+handleCoordinationEvent(event: CoordinationEvent<?>)
+getId(): String
CoordinationEvent<T>
-data: T
-source: String
CoordinationInformation<T>
-sourceEvent: CoordinationEvent<T>
-decoLinkId: String
-senderId:String
fires and receives
fires and receives
sends and receives
sends and receives
0
1
1
0
1
*
0
*

Fig. 4. UML class diagram of the Decentralized Coordination Framework.

component-specific coordination event handling should be implemented. Also, an according coordination endpoint has to created for the component. The framework provides a helper function that creates such a coordination endpoint and connects it to the component with a coordination event bus. The framework provides a ready-to-use implementation of the `CoordinationEndpoint` class as well as a generic implementation of the `CoordinationInformation` class. Abstract super classes exist to implement specific `CoordinationMedium` and `CoordinationEvent` classes. Therefore, if an application should be equipped with coordination capabilities the following steps have to be performed:

1) Writing the XML-based declarative coordination process description that instructs the coordination endpoints.
2) Implementing the `ICoordinatable` interface for the components that should be coordinated.
3) Identifying the relevant coordination events and implementing them using the abstract `CoordinationEvent` super class.
4) Implementing the coordination logic for the information propagation by extending the abstract `CoordinationMedium` super class.

IV. Simulation: Self-Organizing Redistribution of Bikes

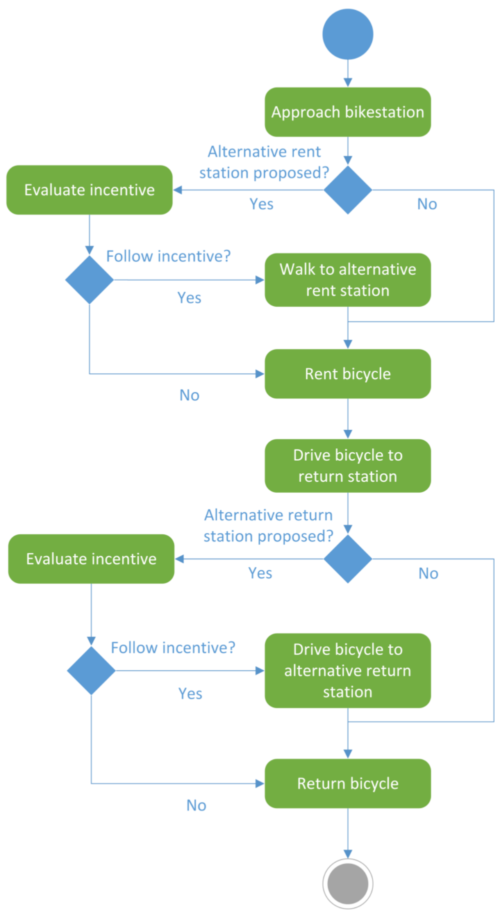
The decentralized coordination framework described in the previous section has been used to implement a coordination process in order to control the self-organizing redistribution of bikes in a simulated bike-sharing system. This section will give an overview about the implementation details of the simulation system as well as the usage of the coordination framework. Also, the simulation scenario will be described.

A. *Implementation Details*

The simulation system is realized using RinSim [6], a logistics simulator written in Java. It supports (de)centralized algorithms for dynamic pickup-and-delivery problems (PDP). The cyclists renting bikes at stations were realized as agents. They are created at a specific station, where they try to rent a bike and if a bike is available at the station, they drive it to their designated destination stations, where they return it. In order to map the road model of Washington, D.C., the corresponding area was extracted from Open-StreepMap (OSM) [28] and transformed into the graph-based road model supported by RinSim. Thus, the movement of the cyclists along the roads can be simulated by moving them on the edges of the resulting graph. Following the PDP-modeling approach, the bikes were modeled as parcels and the bike stations as depots. Time in the simulation system is simulated in a discrete manner, divided into ticks of 60 seconds length. Therefore, the simulation of a whole day consists out of 1440 simulation ticks.

In order to rebalance the availability of bikes at the stations, as a possible addition to the truck-based redistribution efforts ventured by Capital Bikeshare, an incentive scheme for the users to stimulate them to re-distribute bikes in a self-organizing fashion is proposed. The approach is based on the concept that whenever a user tries to rent a bike at an empty or nearly empty station, an alternative rent station with a sufficient amount of bikes is suggested to the user. Equivalent, whenever a user tries to return a bike at a full or nearly full station, an alternative return station with a sufficient amount of free docks is suggested to the user. Thus, the distribution of bikes among the stations will be balanced in a self-organizing way, as users renting a bike are detoured from empty stations to, preferably full or nearly full ones or at least non-empty ones. The same goes for the returning of bikes, where users are detoured from full or nearly full stations to preferably empty or at least non-full ones.

For the simulation the alternative stations are proposed to the user agents whenever they approach a station, in a real world scenario a mobile phone application is imaginable that informs users about alternative rent or return stations before they actually approach them in case they state their intent to approach a station a priori. Possible incentives for a detour are, e.g., free minutes that are added to the users next trips or virtual bonus points that they can exchange for goodies in web shop, similar to the bonus programs of many retail stores. Fig. 5 depicts the whole process from a user's point of view.



Approach bikestation
Alternative rent station proposed?
Evaluate incentive
Yes
No
Follow incentive?
Walk to alternative rent station
Yes
Rent bicycle
No
Drive bicycle to return station
Alternative return station proposed?
Evaluate incentive
Yes
No
Follow incentive?
Drive bicycle to alternative return station
Yes
Return bicycle
No

Fig. 5. UML activity diagram describing the rent/return process from a user's perspective.

A decentralized coordination approach is used to calculate the alternative rent and return stations that are suggested to the users. It is realized using DeCoF. Bike stations periodically send their current occupancy rate to all other bike stations within a certain circular communication range. Stations receiving such status updates from other stations collect them and use them to calculate alternative rent and return stations in a decentralized way. Whenever such status updates are received, the receiving bike station determines the station with the lowest and the highest occupancy rate from the list of stations. The station with the lowest occupancy rate is selected as the alternative return station and the station with the highest occupancy rate is selected as the alternative rent station. These alternative stations are suggested to a user whenever it tries to rent a bike at the station when the station is currently empty, respectively when the user tries to return a bike at the station when it is currently full or critical occupied. So, the maximum detour distance equals the communication range of the stations, as only stations that are within a station's communication range are considered. For the simulated scenarios, a critical occupancy rate of 75% is used.

Following the usage instructions of the DeCoF the bike stations that are implemented as RinSim specific depot agents (`Bikestation`), implement the `ICoordinatable` interface and are equipped with a coordination endpoint. Every minute of the simulated time they fire an `BikeStationStatusUpdateEvent` as a coordination event which contains their current occupancy rate. The according coordination endpoint publishes this as part of a coordination information over a `RoadBasedCoordinationMedium`. This medium extends the abstract `CoordinationMedium` super class with RinSim specific coordination logic. Therefore, it has a reference to the simulator's road model, so it has knowledge about the simulation environment and the position of all the bike stations. The circular communication range is a configurable coordination parameter of the medium. Based on the road model the medium selects all bike stations within the communication range and publishes the coordination information to their according coordination endpoints. When receiving such coordination information the endpoints trigger a `BikeStationStatusUpdateEvent` in the coordinatable bike stations and thus, initialize the calculation of the alternative rent and return stations. Listing 1 shows the declarative coordination process description for this application. It shows how the `Bikestation` agent is mapped to the `bikestation` role. This role is used to instruct a decentralized coordination link (`deco-link`). The link definition contains information about the affected roles, the `bikestation` role in this case, and how the coordination events should be mapped to each other as well as which coordination medium should be used for the information propagation. In this case the `RoadBasedCoordinationMedium` is used and it is configured with a `maxCommRange` coordination parameter limiting the information dissemination. In case of the listing a communication range of 1 km is used.

```xml
<coordination-description context="BikeSharing">
 <role-definitions>
  <role name="bikestation">
   <components>
    <component class="de.haw.c4das.bikesharing.
         simulation.Bikestation" />
   </components>
  </role>
 </role-definitions>
 <deco-link-definitions>
  <!-- Bikestation status update link -->
  <deco-link id="bs-update">
   <from role="bikestation" event="de.haw.c4das.
        bikesharing.simulation.coordination.
```

```
            BikeStationStatusUpdateEvent" />
        <medium class="de.haw.c4das.bikesharing.
            simulation.coordination.
            RoadBasedCoordinationMedium">
         <properties>
          <property key="maxCommRange" value="1" />
         </properties>
        </medium>
        <to role="bikestation" event="de.haw.c4das.
            bikesharing.simulation.coordination.
            BikeStationStatusUpdateEvent" />
      </deco-link>
    </deco-link-definitions>
  </coordination-description>
```

Listing 1. Bikesharing coordination process description (XML).

Fig. 6 visualizes the system's self-organizing dynamic that results from the previous described concepts. The dynamic is composed of two parts, the *Bikestation Coordination* as well as the *User Cooperativeness*, and it results from the interactions between them. The coordination among the bikestations generates the alternative rent or return stations that are proposed to the users. Then the self-organizing dynamic of the systems depends on the users' willingness to follow such a proposal.

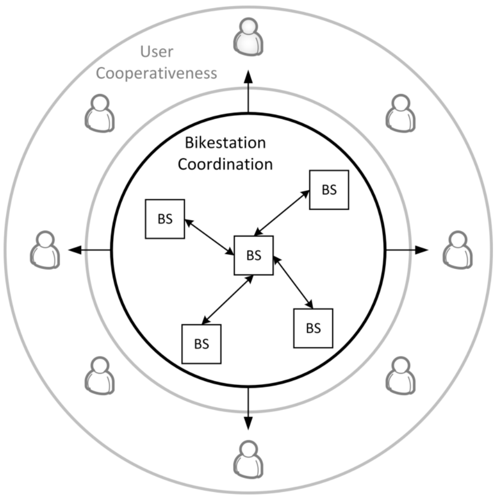


Fig. 6. Self-organizing dynamic of the bikesharing-system.

B. Simulation Scenario

A typical Monday was examined in order to simulate Washington D.C.'s bike-sharing system and to evaluate the impact of the proposed coordination strategy. Therefore, the trip history data of all Mondays (except holidays) from 2014 provided by Capital Bikeshare (the system's operator) was analyzed. To do so, the day was divided into 24 time slices. For each of these 24 time slices the departure probabilities q and the dependent destination probabilities q for the bike-stations were calculated based on the ventured trips: The departure probability p_A for a station A is denoted by

$$p_A = \frac{n_A}{N},$$

with n_A as the number of all trips started at station A while N is the total number of trips. The dependent destination probability q_B is denoted by

$$q_B = \frac{d_B}{D_A}.$$

It is characterized by the fraction of numbers of departures to station B from station A denoted as d_B and the total number of departures from station A denoted as D_A. The simulated scenario starts at 12 a.m. In order to simulate the different rush at different times of the day, the mean total number of trips for each of the 24 hours of the day was determined based on the trip history data. During the execution the simulator generates the number of cyclist agents specified by the rush equally distributed for the currently simulated time slice. As a simplification, all cyclists move with a constant speed along the graph-based road model. In order to find a route from the departure to the destination stations, they use a shortest path approach and traverse the edges of the graph road model, considering the edge weight as the distance to the next node. The simulation was configured to allow an overcrowding of bike stations, when no free docks are available. If a cyclist agent tries to rent a bike at and empty station, this incident is reported and the total number of rides that did not take place is returned as part of the simulation results for evaluation purposes. Fig. 7 shows an extract of the simulated map with the road model and some of the cyclists and bike stations as well as the chosen communication range.

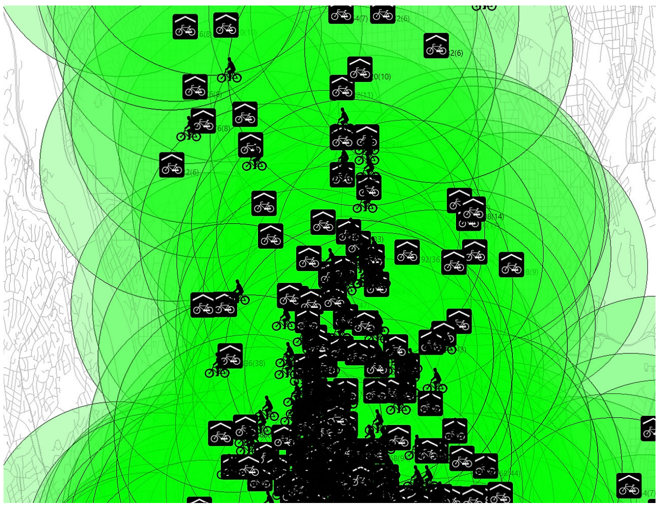


Fig. 7. Extract of the simulated map showing the road model, some of the cyclists and bike stations as well as the chosen communication range.

V. Simulation: Self-Organizing Redistribution of Bikes

In order to evaluate the impact of the self-organizing redistribution strategy depending on the communication range as the relevant coordination parameter, different scenarios with fluctuating communication ranges as well as a reference scenario without any self-organizing behavior were simulated. In all of this scenarios the cyclists moved with a fixed speed of 18km/h. Initially, half of the docks of the bike stations were occupied. The bike stations' number of docks was extracted from the data provided by operator via the *Capital Bikeshare Dashboard* [7]. For all the scenarios simulating the self-organized redistribution of bikes, a user cooperativeness rate of a 100% was assumed. This means that the users always follow a proposed detour to an alternative rent respectively return station. This assumption was made for a more precise evaluation of the impact of the communication range. A detailed analysis of the impact of the users' cooperativeness rate on the proposed self-organizing redistribution strategy is presented in [5].

Fig. 8 shows and compares the results of a simulation scenario with no self-organizing redistribution of bikes and one with a communication range limited to 1,5 km. The parameters of the simulated scenario are summarized in Table I. The figure depicts the number of stations that are in a normal state (neither full nor empty) for both scenarios. It is observable for both cases, how the number of normal stations declines with the morning rush-hour beginning at around 7 a.m. (minute 420). Over the day, these numbers fluctuate only a little. In the late afternoon (around minute 1000) the number of normal stations recovers a bit. This behavior can be explained by the rush-hour movements of commuters. In the morning they drive from the suburbs to the city center and return in the afternoon. Stations in the suburbs tend to become empty during the morning rush-hour, while stations in the city center tend to become full or over-crowded. The returning commuters in the afternoon take bikes from the overcrowded stations in the city center and refill the empty ones in the suburbs when they return, thus increasing the number of normal stations. The figure gives a first impression about how the self-organizing redistribution of bikes improves the number of normal stations over the whole day.

TABLE I
PARAMETERS OF THE CONDUCTED SIMULATION.

Simulation Parameter	Value
No. of cyclists	7433
No. of stations	345
Cyclist speed	18 km/h
Initial bikestation utilization rate	50%
User cooperativeness rate	100%
Bikestation communication range	1.5 km

In order to further measure the impact of the self-organizing redistribution with regards to the maximum communication range, 6 simulations with different communications ranges (from 0,5 km to 3 km) were performed and compared to the reference scenario. Apart from the fluctuating communication range the same simulation parameters as displayed in Table I were used. The according results are shown in Fig. 9. It contains the mean deviation of the number of normal stations in comparison to the previous described reference scenario with no self-organizing behavior. This value states how many more stations in average over the simulated day are in the normal state in comparison to the reference scenario. The figure shows that with a communication range of 1,5 km a

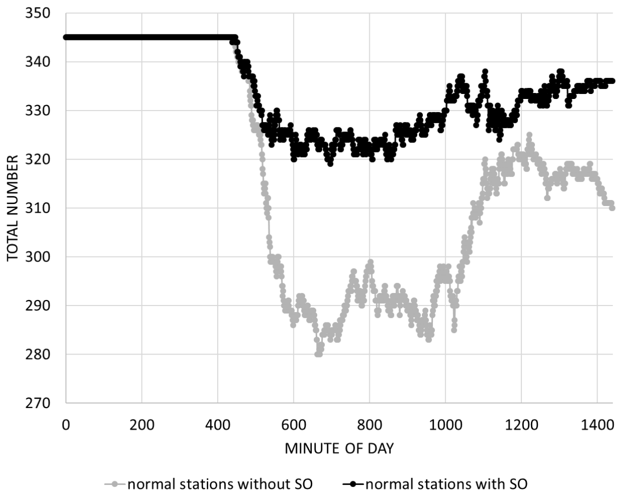


Fig. 8. Results of a simulation scenario with enabled and disabled self-organization.

maximum deviation of 10,7% can be achieved. This means that about 10% more of the total amount of stations are in the normal state in comparison to the scenario with no self-organizing behavior. Considering 345 stations in total, an improvement of 10% of the stations means that about 35 more stations are in the normal state. Fig. 8 shows that without any self-organizing redistribution efforts at least about 280 stations are in the normal state. This results in 65 stations being empty or full/overflown. By increasing the number of normal stations by 10% of all stations to about 315 stations, the number of stations that are empty or full/overflown is reduced by about 55% to 30 stations.

As the communication range is directly related to the distance a user is detoured when he/she wants to rent or return a bike, it is obvious that this value should be as low as possible to increase user acceptance. Also the figure shows that a higher communication range actually may have a negative impact on the results. One potential reason is that the trip time is lengthened and therefore, also the time the bikes are in usage and thus, not available at the stations. In addition, a higher communication ranges may decrease the density of bikes in the area where they are actually needed. For example, they might be detoured from the city center back to the suburbs if the communication range is too high and therefore, no longer available to meet the higher demand in the city center.

VI. Conclusion and Future Work

The acceptance of bike-sharing systems depends heavily on the availability of bikes at the stations. In spite of truck-based redistribution efforts by the operators, stations still tend to become empty or full, especially in rush-hour situations. In this paper, we explored an approach for the self-organizing redistribution of bike by the users and presented a decentralized coordination framework for the realization of self-organizing systems. It is based on the concept that the self-organizing dynamic that causes a system to adapt to external

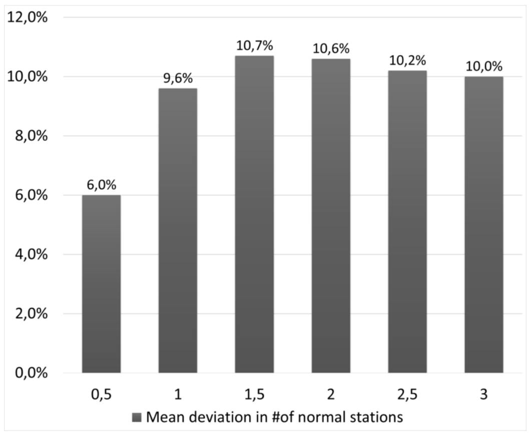


Fig. 9. Mean deviation of the number of normal stations in comparison to a reference scenario with no self-organizing behavior. X-axis denotes communication ranges in km.

and internal influences is controlled by decentralized coordination processes. The processes describe the self-organizing behavior that continuously structures, adapts and regulates aspects of the application. They instruct a set of decentralized coordination media and coordination endpoints. Coordination media deal with the interactions between the components (information propagation), while the coordination endpoints handle the adaptation of the components (local entity adaptation). Together they control the microscopic activities of the components, which on a macroscopic level lead to the manifestation of the intended self-organizing dynamic. The integration of the coordination endpoints and media is prescribed by declarative defined coordination processes which structure and instruct their operations. A microscopic simulation of a bike-sharing system based on data taken from Washington, D.C. (2014) was realized to show how the presented framework can be used to build self-organizing applications. In case of the presented bike-sharing system, a decentralized coordination process was introduced that allowed the bike stations to communicate their current occupancy rate to other stations within a certain circular communication range to calculate alternative rent and return stations for the users. Thereby, we measured and evaluated the impact of the communication range as the relevant coordination parameter for the performance of the self-organizing behavior.

Future work will deal with the structural adaptation of coordination processes. Under certain conditions self-adaptive systems may exhibit a behavior where performance decreases and/or starvation may occur. Structural adaptations are a promising concept under such conditions, that can be used to realize the dynamic exchange or reconfiguration of self-organizing coordination processes. By facilitating the concept of structural adaptations presented in [29] for the proposed decentralized coordination framework it is envisioned to realize the dynamic exchange of the described coordination process. Thereby, another coordination process based on the formation of clusters among the stations, where a random station will take on the role of a super station for this cluster and receive status updates from all stations within the cluster, so it can calculate the optimal alternative rent and return stations for all stations within the cluster, will dynamically replace the existing coordination process at runtime.

REFERENCES

[1] P. Vogel and D. Mattfeld, "Modeling of repositioning activities in bike-sharing systems," in *Transport Research (WCTR), 2010 World Conference on*, 2010.

[2] P. DeMaio, "Bike-sharing: History, impacts, models of provision, and fu-ture," *Journal of Public Transportation*, vol. 12, no. 4, pp. 41–56, 2009.

[3] S. Bührmann, "Bicycles as public-individual transport - european developments." Rupprecht Consult Forschung und Beratung GmbH, Cologne, Germany, Tech. Rep., 2008.

[4] P. Midgley, "The role of smart bike-sharing systems," *Urban Mobility Journeys*, vol. 2, pp. 23–31, 2009.

[5] T. Preisler, T. Dethlefs, and W. Renz, "Data-adaptive simulation: Cooperativeness of users in bike-sharing systems," in *Logistics (HICL), 2015 Hamburg International Conference of*, W. Kersten, T. Blecker, and C. M. Ringle, Eds., vol. 20. epubli GmbH, 2015, pp. 201–228.

[6] R. van Lon and T. Holvoet, "Rinsim: A simulator for collective adaptive systems in transportation and logistics," in *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, Sept 2012. doi: 10.1109/SASO.2012.41. ISSN 1949-3673 pp. 231–232.

[7] "Capital bikeshare dashboard," http://cabidashboard.ddot.dc.gov, accessed: August 4, 2016.

[8] M. Martinez, "Washington, d.c. launches the nation's largest bike sharing program," *Grist Magazin*, 2010.

[9] J. Maus. (2013) Behind the scenes of capital bikeshare. Available at: http://bikeportland.org/2013/03/10/behind-the-scenes-of-capital-bikeshare-84006 (retrieved August 4, 2016).

[10] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *Artifical Intelligence (IJCAI), 2009 International Joint Conference on*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1420–1426. [Online]. Available: http://dl.acm.org/citation.cfm?id=1661445.1661673

[11] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, "Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 455–466, Aug. 2010. doi: 10.1016/j.pmcj.2010.07.002. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2010.07.002

[12] L. Zhang, J. Zhang, Z. yu Duan, and D. Bryde, "Sustainable bike-sharing systems: Characteristics and commonalities across cases in urban china," *Journal of Cleaner Production*, vol. 97, pp. 124–133, June 2015.

[13] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[14] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck, "Organic computing - addressing complexity by controlled self-organization," in *Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), 2006 International Symposium on*, ser. ISOLA '06. IEEE Comp. Soc., 2006. ISBN 978-0-7695-3071-0 pp. 185–191.

[15] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Software engineering for self-adaptive systems through feedback loops," B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Engineering Self-Adaptive Systems through Feedback Loops, pp. 48–70. ISBN 978-3-642-02160-2

[16] J.-P. Mano, C. Bourjot, G. A. Lopardo, and P. Glize, "Bio-inspired mechanisms for artificial self-organised systems," *Informatica (Slovenia)*, vol. 30, no. 1, pp. 55–62, 2006.

[17] D. Weyns, S. Malek, and J. Andersson, "On decentralized self-adaptation: Lessons from the trenches and challenges for the future," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2010 ICSE Workshop of*, ser. SEAMS '10. New York, NY, USA: ACM, 2010. doi: 10.1145/1808984.1808994.

ISBN 978-1-60558-971-8 pp. 84–93. [Online]. Available: http://doi.acm.org/10.1145/1808984.1808994

[18] J.-Y. Jung, J. Park, S.-K. Han, and K. Lee, "An eca-based framework for decentralized coordination of ubiquitous web services," *Information and Software Technology*, vol. 49, no. 11-12, pp. 1141 – 1161, 2007. doi: http://dx.doi.org/10.1016/j.infsof.2006.11.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584906001959

[19] E. Nardini, M. Viroli, M. Casadei, and A. Omicini, "A self-organising infrastructure for chemical-semantic coordination: Experiments in tucson," in *Proceedings of the 11th WOA 2010 Workshop, Dagli Oggetti Agli Agenti, Rimini, Italy, September 5-7, 2010.*, 2010. [Online]. Available: http://ceur-ws.org/Vol-621/paper17.pdf

[20] D. Gelernter and N. Carriero, "Coordination languages and their significance," *Commun. ACM*, vol. 35, no. 2, pp. 97–107, 1992. doi: 10.1145/129630.129635. [Online]. Available: http://doi.acm.org/10.1145/129630.129635

[21] Z. Li and M. Parashar, "A decentralized agent framework for dynamic composition and coordination for autonomic applications," in *Database and Expert Systems Applications, 2005 Sixteenth International Workshop on*, Aug 2005. doi: 10.1109/DEXA.2005.10. ISSN 1529-4188 pp. 165–169.

[22] T. Preisler, T. Dethlefs, and W. Renz, "Middleware for constructing decentralized control in self-organizing systems," in *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, July 2015. doi: 10.1109/ICAC.2015.56 pp. 325–330.

[23] A. Pokahr and L. Braubach, "The active components approach for distributed systems development," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 28, no. 4, pp. 321–369, 2013. doi: 10.1080/17445760.2013.785546. [Online]. Available: http://dx.doi.org/10.1080/17445760.2013.785546

[24] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-organization in multi-agent systems," *The Knowledge Engineering Review*, vol. 20, no. 2, pp. 165–189, Jun. 2005. doi: 10.1017/S0269888905000494. [Online]. Available: http://dx.doi.org/10.1017/S0269888905000494

[25] M. Prokopenko, "Design vs. self-organization," in *Advances in Applied Self-organizing Systems*, ser. Advanced Information and Knowledge Processing, M. Prokopenko, Ed. Springer London, 2008, pp. 3–17. ISBN 978-1-84628-981-1. [Online]. Available: http://dx.doi.org/10.1007/978-1-84628-982-8_1

[26] J. Sudeikat and W. Renz, "Decomas: An architecture for supplementing mas with systemic models of decentralized agent coordination," in *Web Intelligence and Intelligent Agent Technologies (WI-IAT), 2009 IEEE/WIC/ACM International Joint Conferences on*, vol. 2, Sept 2009. doi: 10.1109/WI-IAT.2009.137 pp. 104–107.

[27] A. S. Rao, M. P. Georgeff *et al.*, "Bdi agents: From theory to practice." in *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, 1995, pp. 312–319.

[28] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing, IEEE*, vol. 7, no. 4, pp. 12–18, Oct 2008. doi: 10.1109/MPRV.2008.80

[29] T. Preisler, W. Renz, and T. Dethlefs, "Structural adaptation for self-organizing multi-agent systems: Engineering and evaluation," *International Journal on Advances in Intelligent Systems*, vol. 8, no. 3&4, pp. 413–425, 2015.