Handbook of Computational Social Science - Vol 2
Edited By Uwe Engel, Anabel Quan-Haase, Sunny Xun Liu, Lars Lyberg

Large-scale Agent-based Simulation and Crowd Sensing with Mobile Agents

Stefan Bosse

Stefan Bosse

**University of Bremen, Dept. of Mathematics & Computer Science,
28359 Bremen, Germany**

Abstract. *The main topic of this chapter is the introduction of the concept of the fusion of real and virtual worlds creating augmented virtuality by using mobile agents and agent-based simulation technologies. Virtual worlds are created by simulation or by digital games. Socio-technical systems are characterised by interactions of (1) Human-to-Human (initiated by a human) (2) Human-to-Machine (initiated by a human) (3) Machine-to-Human (initiated by a machine, e.g., a chat bot) and (4) Machine-to-Machine (initiated by a machine) interfaces. Modelling of such complex interaction networks is difficult. The simulation of social ensemble behaviour requires simplification of interactions and individual behaviour. Commonly simulations are performed with a number of entities (humans, machines, ..) in a sandbox world signifantly below real population sizes. Agent-based Modelling (ABM) is a suitable behaviour model for simulation. Simulation worlds are commonly closed and rely on artificial sensory information generated by the simulator program or using data collected off-line (→ field studies). A new simulation paradigm providing augmented virtuality is introduced providing the coupling and integration of Mobile Crowd Sensing and social Data Mining in agent-based simulation worlds. The simulation world interacts with real world environments, humans, machines, and other virtual worlds in real-time using agent-based Modelling and Computation. Agents can represent artificial humans, bots, machines, and agents can be used for distributed mobile computing, e.g., Crowd Sensing. The concept of agent-based virtual sensors and sensor aggregation is introduced with examples from Crowd Sensing applications.*

1.1 Introduction

The investigation of socio-technical aspects in society, business, and industry requires accurate models and tools to test the models for validity and conformance to real world phenomena. Commonly, field studies (surveys) and simulations are used to study effects of behaviour and networking models on populations and to determine model parameters. Field studies are performed in real world environments, whereas simulation is performed in a closed-box virtual world.

Agent-based methods are established for modelling and studying of complex dynamical systems and for implementing distributed intelligent systems, e.g., in traffic and transportation control (Hamidi et al., 2018) (Wang, 2005). Therefore, agent-based methods can be divided into five main classes (Baqueiro et al., 2008):

1. Agent-based Modelling (ABM) - Modelling of complex dynamic systems by using the agent behaviour and interaction model $\Rightarrow$ **Physical agents**
2. Agent-based Computing (ABC) - Distributed and parallel computing using mobile agents related to mobile software processes $\Rightarrow$ **Computational agents**
3. Agent-based Simulation (ABS) - Simulation with physical agents to simulate distributed multi-entity systems or simulation of computational agents
4. Agent-based Modelling and Simulation (ABMS) - the agent model is part of the study resulting in a close modelling and simulation using the agent behaviour and interaction model
5. Agent-based spatial modelling and simulation (ABSS) - extending the simulation world with spatial data and Geographical Information Systems

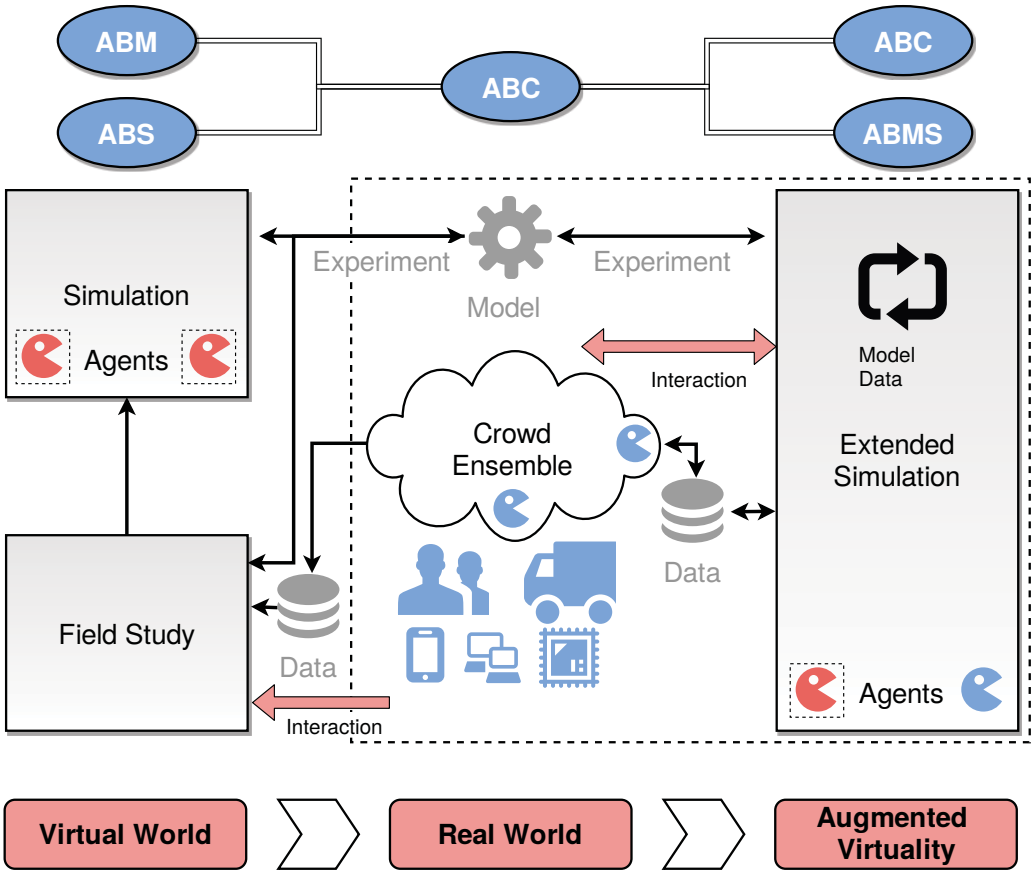

Fig. 1. (Top) The four different agent-based methods with overlapping relations - ABM: Agent-based Modelling, ABC: Agent-based Computing, ABS: Agent-based Simulation ABMS: A. Modelling and Simulation (Bottom) Transition from separated field studies and simulation towards extended simulation merging real and virtual worlds

Two promising fields of current studies in computer science are Data Mining (DM) and Agent-based Modelling and Simulation (ABMS) (Baqueiro et al., 2008). Agent-based simulation is suitable for modelling complex social systems with respect to interaction between individual entities, manipulation of the world, spatial movement, and emergence effects of groups of entities. The main advantage is the bottom-up modelling approach composing large-scale complex systems by simple entity models. The main disadvantage of ABM is the (over-) simplified entity behaviour and simplification of the world the entities acting in. Commonly, simulation

bases on synthetic data or data retrieved by field studies. Many simulations and models lacking of diversity existing in real world. Commonly, sensor and model data (parameters) used in simulations (virtual world) is retrieved from experiments or field studies (real world), shown in Fig. 1. But there is neither a feedback from the virtual to the real world nor an interaction of the real world with the virtual world and vice versa.

ABS and ABMS is commonly constructed using collections of condition-action rules to be able to percept and react to their situation, to pursue the goals they are given, and to interact with other agents, for example, by sending them messages (Gilbert, 2004). The *NetLogo* simulator is an example for an established ABS tool used in social and natural sciences (Wilensky et al., 2015), but is limited to behavioural simulation only (ABM domain). In this work, a different simulation approach combining ABM, ABC, and ABS methodologies, is used deploying the widely used programming language JavaScript. JavaScript is used increasingly as a generic programming language for ABC and ABS (Bosse et al., 2018) (Calenda et al., 2016).

Mobile devices like smart phones are valuable sources for social data (Ganti et al., 2011), either by participatory crowd sensing with explicit participation of users providing first class data (e.g., performing surveys or polls) or implicitly by opportunistic crowd sensing collecting secondary class data, i.e., traces of device sensor data delivering, e.g., actual position, ambient conditions, network connectivity, digital media interaction, and so on. Crowd sensing and Social Data Mining as a data source contribute more and more to investigations of digital traces in large-scale machine-human environments characterised by complex interactions and causalities between perception and action (decision making).

But mobile devices are not limited to being sensors. They can be used as actuators, too, by interacting with the user via chat dialogues or social media that can affect the user behaviour (e.g., mobility decisions, consumer control). Moreover, mobile networks can be coupled with device and machine networks (i.e., the Internet of Things) creating augmented data and enabling advanced spatial and contextual tracking.

It is difficult to study such large-scale data collection, data mining, and their effect on societies, domestic services, and social interaction in field studies due to a lack of reliable data and complexity. Agent-based modelling of socio-technical systems is well established (van Dam et al., 2013) and can be applied for smart city management, however commonly ap-

plied in an artificial world., i.e., a simulation is performed in virtual reality worlds only to derive and proof models under hard limitations. In this work, a new concept and framework for augmented virtual reality simulation is introduced, suitable, but not limited to, to investigate large-scale socio-technical systems. Mobile agents are used already successfully in-field crowd sensing (Leppäne et al., 2017). In this work, mobile agents are used to combine in-field ubiquitous crowd sensing, e.g., performed by mobile devices, with simulation.

A chat bot agent (i.e., a dialogue robot) is capable to perform dynamic and situation-aware dialogues with humans to get empirical data into the simulation and to propagate synthetic data from the simulation world into the real world. The chat bot can act as an avatar providing information for users, e.g., for optimised and dynamic traffic control based on real (covering actual and history data) and simulated data (addressing future predictions).

The chat bot agents (as computational agents) can operate both in real and virtual worlds including games and providing a fusion of both worlds by seamless migration. Agents are loosely coupled to their environment and platform and interact with each other via tuple spaces (generative communication) and via uni-cast or broadcast signals. Mobility is provided by agent process snapshot migration between platforms.

The novel MAS crowd sensing and simulation methodology, introduced in the next sections, is suitable to combine social and computational simulations with real-world interaction at run-time and in real-time, e.g., by integrating crowd sensing, using mobile agents. There are two classes of mobility that can be modelled in the simulation: (1) Social mobility on short- and long-term time scales (2) Plan-driven traffic mobility.

The Crowd Sensing performed by mobile agents can be used to create digital twins of real humans (with respect to the social and technical interaction model and mobility) based on individual surveys and user information collected via a chat bot dialogue.

Among the capability of real-time simulation, the simulation framework is capable to create simulation snapshot copies that can be simulated in parallel in non real-time to get future predictions (simulation branches) from actual simulation states that can be back propagated into the current real-time simulation world.

The following sections introduce the agent-based methodologies addressing augmented simulation and mobile crowd sensing. Fundamental concepts enabling the augmentation of simulation is the concept of digital twin derived from survey data (off- and on-line) and the concept of physical and computational agents. Finally, two demonstrations and use-cases are provided to show the power of the agent-based augmented simulation approach coupling real and virtual worlds.

1.2 Agent Models and Architectures

Basically three different agent behaviour model and architecture classes can be distinguished:

1. Pure functional reactive agents
2. State-based reactive agents
3. Deliberative and knowledge-based agents, i.e., corresponding to the well known Believes-Desires-Intention (BDI) architecture

The second class of state-based reactive agents is close to well known programming models and can be found in ABM, ABC, and ABS domains. The *NetLogo* agent model belongs to this class, too. In the next section, the unified agent model used in this work is shortly introduced and discussed.

1.2.1 ATG Behaviour Model

Details of the unified agent model can be found in (Bosse, 2016), and details about the used agent processing platform *JAM* can be found in (Bosse et al., 2018).

To summarize, the agent behaviour model is purely reactive and state-based, shown in Fig. 2. An agent consists of code and private data (body variables). The code describes the agent behaviour consisting of activities executing actions. Typical actions are:

- Agent creation and termination
- Agent replication
- Agent behaviour modification (changing activities and/or transitions)
- Agent mobility
- Agent communication

- Computation

There are conditional and unconditional transitions between activities. The conditions access agent body variables only. Both code and data are mobile and an agent process snapshot is capable to migrate between two agent platforms. Activities of an agent represent intentions and micro goals, e.g., changing the spatial position, modifying the environment, communicating with other agents, and agent replication. Agents processes support the concept of control path blocking, i.e., the agent processing can be suspended for waiting for an event or the satisfaction of a constraint condition. Replication creates either copies of the generating agent (forking) or new agents instantiated by an agent class.

In contrast to commonly used reactive agent behaviour models, the ATG can be modified by the agent itself offering self-adaptivity. An agent can remove or add activities and transitions, either providing sub-classing (specialisation) or learning.

Agents can communicate via tuple spaces (data driven) providing generative communication and by using signal messages (agent driven), basically used by parent-child agent groups. Physical and computational agents can communicate via signals to synchronise or to exchange data.

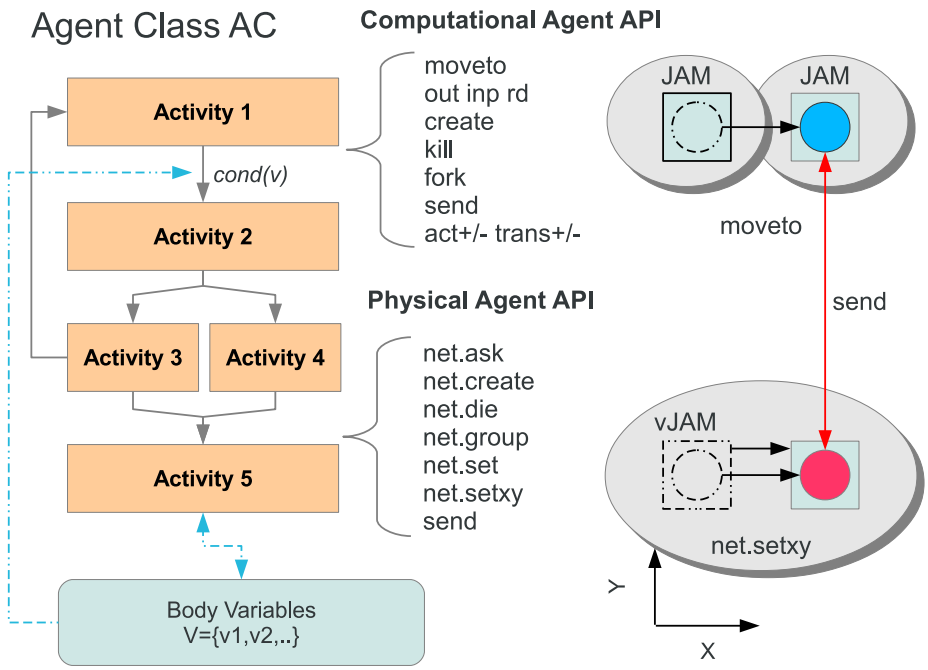

Fig. 2: (Left) Activity-Transition Graph (ATG) behaviour and data model of an agent for a specific class AC. (Middle) Physical and computational agents differ in their action set (Right) Migration and communication between physical and computational agents.

1.3 Augmented Virtuality

There is a significant difference between traditional closed-world simulations and simulations coupled with real world and real-time environments (so called human-in-the-loop simulation). Closed simulations are performed on a short time-scale with pre-selected use cases and input data. The simulation can be processed step-wise without a relation to a physical clock. In contrast, open-loop simulation requires continuous simulation on a large time-scale creating Big Data volumes. A relation to a physical clock is required, too.

The spatial mapping of physical on virtual worlds and vice versa is another issue to be handled. There are basically three possible scenarios and

world mapping models (illustrated in Fig. 3):

1. The real and virtual worlds are isolated (no agent/human/entity of one world know from the existence of humans and entities in the other world)
2. The real world is mapped on the virtual simulation world (simulating the real world with agents representing real and artificial humans or entities)
3. The virtual world extends the real world, e.g., by a game world, and real and virtual entities are aware of each other and can interact with each other, e.g., by smartphone software.

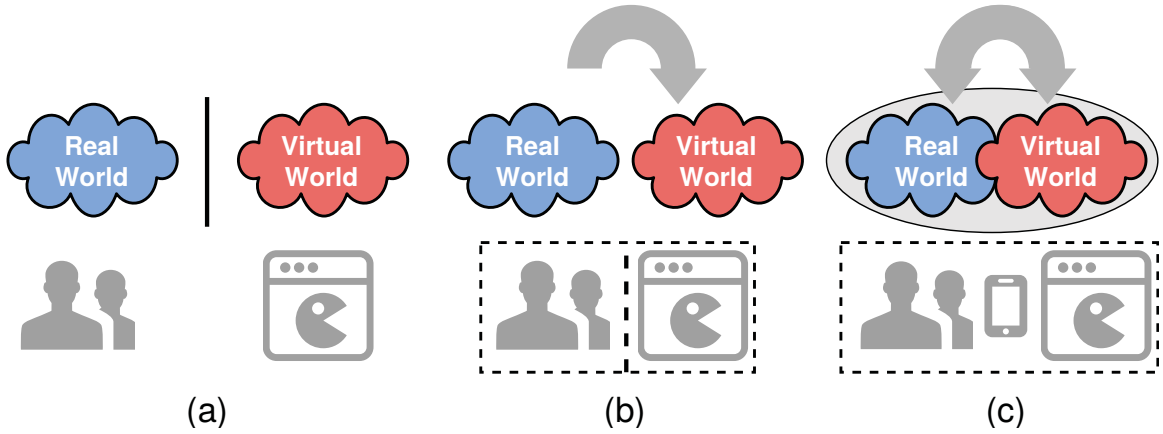

Fig. 3: (a) Real world only deployed with humans separated from virtual world (b) Non-overlapping real and virtual worlds (c) Overlapping real and virtual worlds

In this work, real worlds can be coupled in real-time (1) unidirectional $\mathbb{R} \Rightarrow \mathbb{V}$ by crowd sensing only (2) bidirectional $\mathbb{R} \Leftrightarrow \mathbb{V}$ by crowd sensing and agent-based communication interacting with technical devices (e.g., traffic signs or street lights).

1.4 Virtual Sensors, Big Data, Data Mining, and Digital Traces

According to (McGrath et al., 2014), a large-scale sensing application is

composed of different horizontal domain layers:

1. Sensing and Perception

2. Aggregation

3. Applications

Each horizontal layer is deployed with the following vertical processing layers:

1. Communication

2. Data Processing

3. Security

4. Storage

Although, the first vertical domain layer is represented mostly by physical sensors, all the layers can be represented by virtual sensors. A virtual sensor is a software component acting as a sensor data aggregator. Each sensor component is treated as a sensor, processing an input stream and computing an output data stream. Each physical sensor (data source) is a "data stream" transformer, too, but based on physical principles. In this work, user provided first-class data is considered as physical sensors, too. A virtual sensor is a processing system as well as a data storage (data base). In this work, virtual sensors are represented by mobile agents, performing the sensing, aggregation, and application (or delivery) of accumulated and processed sensor data. The mobility enables self-organising and adaptive mining systems controlled by environmental constraints rather than by individual users. In (McGrath et al., 2014), users using a smart phone App. are considered as agents. This role is replaced in this work by the deployment of agents that perform tasks autonomously.

A virtual sensor consists of different components. The environment of a sensor is a set of input streams of data generated from physical or virtual sensors. The environment defines the context within the virtual sensor operates. An aggregator processes the input streams and performs sensor data fusion, shown in Fig. 4 (left). A filter produces a set of output streams, i.e., generated by replication of agents. Virtual sensors can be coupled in graph-like virtual networks via agent interaction (using signal messages and tuple exchange), shown in Fig. 4 (right). These virtual sensor networks compose processing chains (user defined or ad-hoc and self-organising). Virtual sensors can operate in real-world environments (e.g.,

executed on mobile devices) or in virtual worlds (simulation) and are commonly related to a local context (spatial, temporal, and situation context). Agents implement the sensor data aggregator and filter function, performing the fusion, storage, and communication and represent mobile transport entities with distributed data-directed information processing. Agent processing is virtualised by an unified Agent Processing Platform (e.g., JAM, see Sec. 1.9).

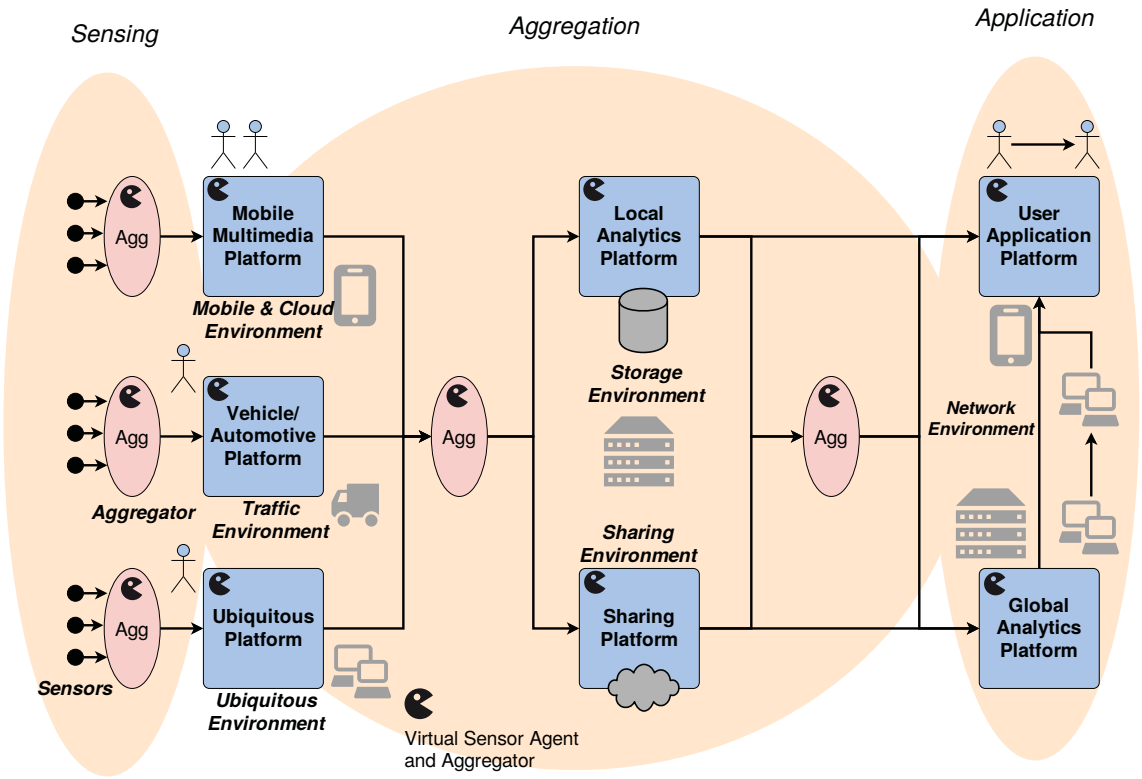

Fig. 4. Mobile Agents as virtual sensors performing sensor fusion (aggregation) connecting platform networks of virtual sensors and sensing, aggregation, and application software layers

Digital trace data are not error-free. Even unobtrusive measures may suffer from measurement error and bias. If subjects are aware of being research subjects, this influences their communication and interaction behaviour, including answer behaviour in surveys. I.e., surveys and Crowd Sensing are reactive and state-based; subjects react to being researched. On the other hand, Big Data means the collection of data already generated.

Although, users are not aware of sampling this kind of data, the data is noisy and biased and highly uncorrelated.

Crowd data can be considered as natively occurring data in contrast to data from surveys and experiments and the typically associated biases through experimental effects and group effects. Computational analyses of big data offer a welcome counterpoint and potential triangulation of multi-method confirmation of key findings in experiments and surveys.

Therefore, sensor data aggregation, filtering, and fusion by such a virtual sensor architecture is vital for meaningful information derivation from large data sets collected from untrusted devices and users.

From the social science perspective, concerns about the reliability and validity of measurements have been raised in various critical papers on Big Data research. Although this is beyond the scope of this work, these aspects have to be addressed already on the technology and information processing level. Among the most frequently discussed issues are (1) comparatively shallow measures, (2) lack of context awareness, and (3) a dominance of automated methods of analysis (Mahrt et al., 2013).

Despite the framework proposed in this work, these relevant issues have to be addressed by further investigations using and applying the agent-driven augmented virtual reality approach dealing with Big Data and meaningful and trustful analysis. Digital traces annotating data with additional information (e.g., trust probabilities, class of data, data source, reliability, proofs) generated by physical and virtual sensors is essential for robust Data Mining and decision making. Agents are suitable to record the data collection, interpreting and assessing the data immediately (at location), and to change data exploration strategies on environmental perception dynamically.

1.5 Workflow

The following Fig. 5 shows the principle work and data flow of the proposed architecture, consisting of agent-based modelling (a formal mode or a software module for modelling the behaviour physical agents) as input for the the agent-based simulation framework, a data base containing generated input and output of the simulation, Data Mining (DM) modules for deriving information from statistical weakly uncorrelated data (including ML), and Crowd Sensing components. The data base is continuously updated during simulation and sensing.

The ABMS relies on generated data stored in a data base provided by an initial data set and data sets derived by DM with sensor data from Crowd Sensing. The Crowd Sensing is performed by mobile chat-bot agents (computational agents). The simulation uses data from the data base as input and produces newly generated output data to the data base. The data flow between the real and virtual worlds is bidirectional, and agents can carry data generated by the simulation to mobile devices and users in the real world.

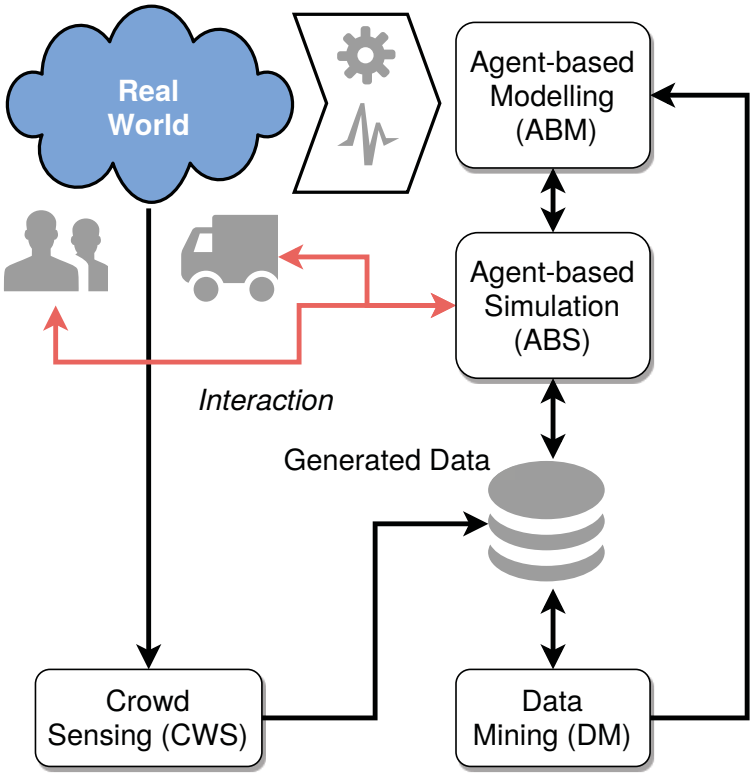

Fig. 5. Principle work and data flow integrating agent-based crowd sensing in agent-based modelling and simulation

1.6 Crowd Sensing and Surveys

The basic crowd sensing and survey architectures can be classified in:

1. Centralised crowd sensing architecture with a single master instance (crowd sourcer)

2. Decentralised crowd sensing architecture with multiple master crowd sourcer instances
3. Self-organising and ad-hoc crowd sensing architecture without any crowd sourcer master instances

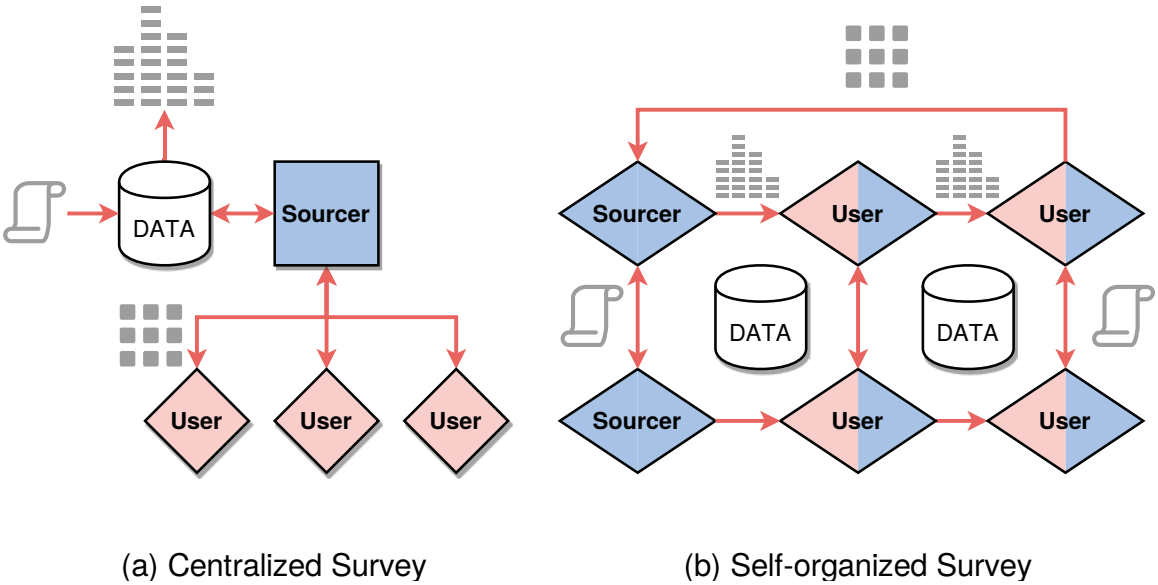

Fig. 6. Different crowd sensing and survey architectures and strategies with data sourcer and users providing data. (a) Centralized architecture (b) Decentralized and self-organized architecture with changing roles of the participants (sourcer/user of surveys and data)

The first class ist the typical WEB-based survey architecture. The surveys are performed with static or dynamic forms contained in WEB pages. There is typically a negotiation between the sourcer and user or the users participating the survey are selected randomly from register data bases.

The survey architecture introduced in this work relates to the second de-centralized approach, although it can be extended with a decentralised and self-organised approach, too, handled by the same framework and mobile chat bot agents.

Among classical data collection the survey aims to create digital twins in the virtual world from survey participants in the real world by deriving twin behaviour model parameters from the survey answers and auxiliary

sensor data (e.g., spatial position), discussed in the next section. The survey is performed by chat bot agents that can perform dynamic dialogues based on answers and context.

In this work, mobile computational agents are used to perform self-organised mobile crowd sensing for survey participation primarily on mobile devices using and chat bot agents.

The crowd sensing via mobile chat bot agents enable the interaction of real humans with agents and digital twins in the simulation world in real-time and vice versa. The digital twins as well as the artificial physical agents in the simulation can interact by dynamically created (influenced) dialogues reflecting the state of the simulation world.

The survey performed by chat bot agents (computational agents) aims to create digital twins in the virtual world from survey participants in the real world by deriving twin behaviour model parameters P from the survey feedback answers F retrieved by a user dialogue D. The survey is performed by chat bot agents that can execute dynamic dialogues via a chat dialogue platform based on previous answers and context.

Mobile agents, e.g., chat bots, can be used to carry information from one location to another. This is typically achieved by transferring a process snapshot of the agents (including data and control state of the agent) between agent processing platforms and computing devices. This technique is discussed in Sec. 1.10.

Moreover, mobile devices deployed with an agent processing platform and carrying mobile agents can be used to collect, carry, and distribute data within large regions via tuple space data bases without direct communication connectivity, shown in principle in Fig. 7. The virtual simulation world is just another region in the mobility regions of agents, and chat bots performing crowd sensing can migrate between real and virtual worlds seamlessly

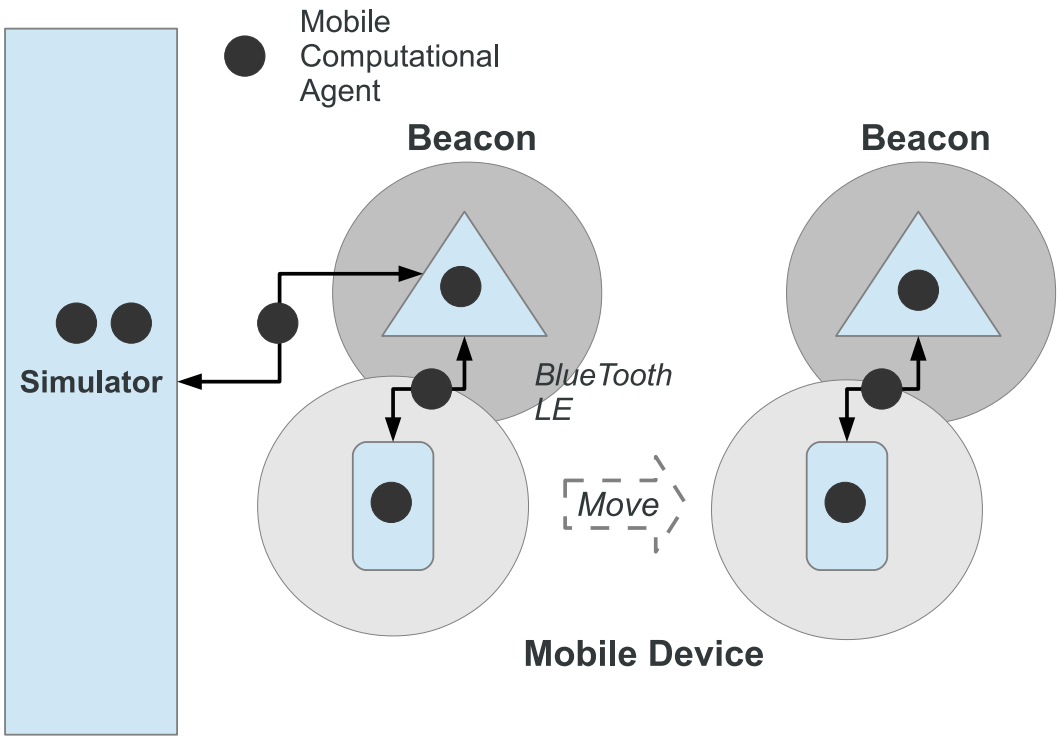

Fig. 7. Computational agents can migrate between simulation and real-world hardware via beacons. Mobile devices can carry computational agents and extend the spatial interaction range of agents. Agents can change the mobile host devices via beacons, too.

1.7 Digital Twins

Commonly, simulation is performed with artificial agent models derived from theoretical considerations or experimental data. Augmented virtuality enables dynamic simulations with agents representing real humans (or crowds). By using Crowd Sensing it is possible to create digital twins of real humans based on a parameteriseable behaviour and interaction model. The parameters of artificial humans in the simulation represented by physical agents are collected by sensor data, i.e., surveys optionally fusioned with physical sensors like GPS, performed by computational exploration and chat bot agents. One simple example is shown in Sec. 1.13.

Physical agents can be equipped with a mental state based on a knowledge base enabling reasoning and deduction. The digital twins as well as the artificial physical agents in the simulation can interact with humans in the real world by dynamically created (influenced) dialogues reflecting the state of the simulation world.

The framework utilising the creation of digital twins, shown in Fig. 8, is capable to simulate different environments in quasi-experimental setups. Controlling all aspects of the virtual world allows to explain the source of structural variation in human interactions, i.e., tackle the hard-to-observe mechanisms of micro-macro interactions.

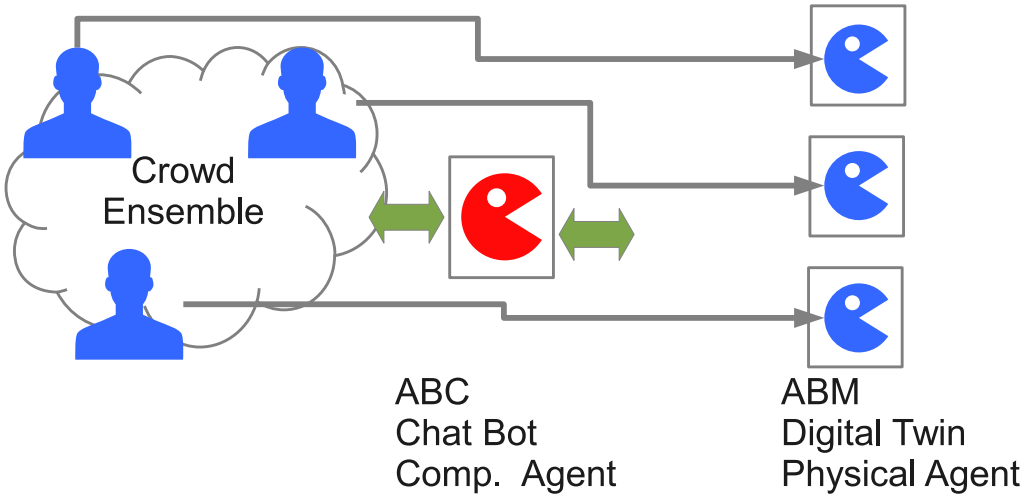

Fig. 8. Creation of digital twins via chat bot agents from real humans

A general agent behaviour model maps the environment state set E by perception P, state I, and knowledge D on plans and actions A, and can be

composed of the following functions:

$$
\begin{aligned}
&ag : E \rightarrow A \\
&see : E \rightarrow P \\
&belief(\Psi) : \Psi \times P \times D \rightarrow D \\
&next(\Psi) : \Psi \times I \times D \times P \rightarrow I \\
&action(\Psi) : \Psi \times I \rightarrow A
\end{aligned}
\tag{1}
$$

The parameter set Ψ=$\{p_1,p_2,..,p_i\}$ of the behaviour model (e.g., social interaction, social expectation, motivation for migration) determines individual behaviour and is derived from real humans via the chat bots. *D* ate

1.8 Physical and computational Agents

In ABM, agents represent natural and physical entities, i.e., humans, animals, machines, all posing some physical interaction. The physical agents spawn a network graph with nodes representing the natural entities and their behaviour. The edges of the network graph represent social and physical interaction. In contrast, in ABC, agents represent software used for computing. These agents spawn a network graph consisting of nodes representing computers and edges representing digital communication.

The approach of augmented virtuality integrates both kinds of agents in one unified agent environment.

1.8.1 Physical Agents

A physical agent is characterised by a behaviour consisting of social and world interaction, mobility, replication and evaluation, and cognitive capabilities. A physical agents (except robots) can only exist in simulation worlds with an abstraction of the world and the physical agent behaviour. A physical agent requires a body, i.e., its own agent platform. Mobility of physical agents means mobility of the body platform carrying the agent. Physical agents can create computational agents, e.g., chat bot agents, originally bound the physical agent platform node, but capable to migrate to other platform nodes.

1.8.2 Computational Agents

A computational agent is characterised by a program that performs computation, access to physical sensors, data exchange and synchronisation with other agents, mobility, replication, and learning. A computational agent is a distributed computing paradigm and can exist (and can be exe-

cuted) in real and virtual simulation worlds. Computational agents require an execution platform. Mobility of computational agents means the migration of a process snapshot of the agent program. But the agent platform can be mobile itself, e.g., considering mobile host devices like smartphones.

Interaction of computational agents bases primarily on synchronised data exchange, e.g., using tuple spaces of signals.

Computational agents can be created by physical agents, shown in Fig. 9. The migration of a computational agent from a node of a physical agent to another platform (either a platform executing computational agents only or a body platform of another physical agent) requires communication ports, e.g., virtual wireless communication channels. Only nodes with overlapping communication ranges can interact with each other, i.e., computational agents are able to migrate between these nodes or communicate with each other remotely.

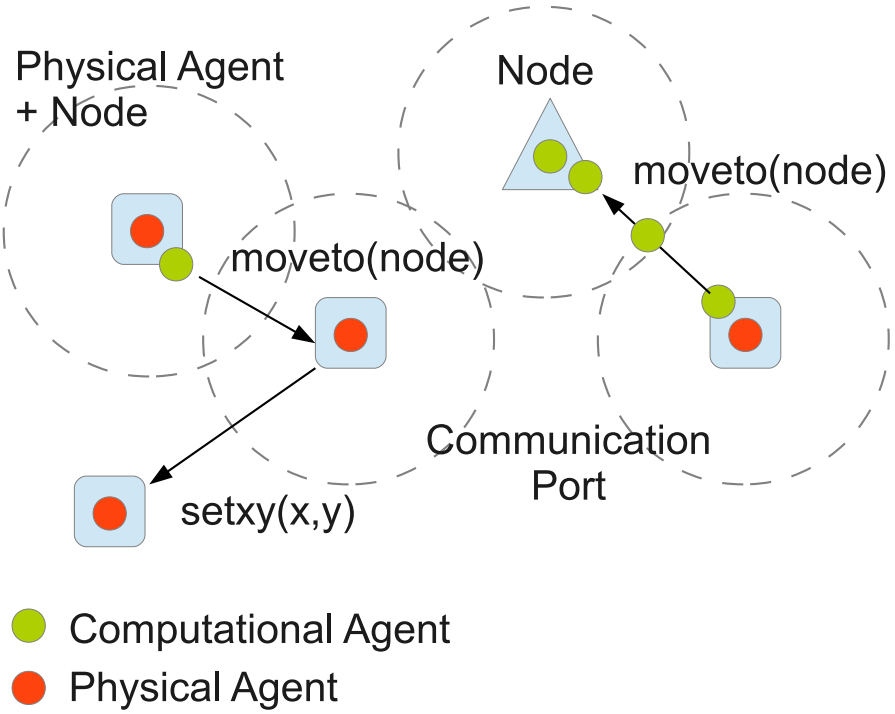

Fig. 9. Physical and computational agents acting in one world

1.9 Agent-based Simulation Software

The following features of a simulation environment for augmented virtuality are required:

1. The simulation environment has to be capable to model and execute physical and computational agents with a unified programming model.
2. The simulation environment requires a network communication interface connecting to the Internet for the migration of computational agents from the virtual in the real world and vice versa.
3. The simulation environment has to provide unified virtual (logical) agent processing platforms for the processing of physical and computational agents.

The *NetLogo* simulation (Wilensky et al., 2015) is well known and established for modelling and simulating societies of physical agents based on a discretised patch grid world, agents with visual properties, and set iterations. But it lacks the integration of computational agents that can interact with physical agents and the real world (i.e., humans via the Internet).

The migration of computational agents requires efficient process checkpointing creating code and data snapshots that can be transferred via a communication link from one to another agent processing platform. The state snapshot must contain:

- The control state (instruction pointer)
- The private data state (body variables)
- The agent behaviour code
- Environmental state information

The snapshot creation requires serialisation and deserialisation of the code and the data in host platform independent byte code format or textual strings. Compiled code shows a strong dependency on the platform and there is commonly no direct code-text conversion available (requiring disassembler and assembler). Interpreted languages avoid these issues, but commonly lacking execution performance. There are basically two exceptions: JavaScript and Lua. Both languages are always parsed on text level, commonly compiled partially to byte code, and finally processed with an additional Just-in-Time (JIT) compiler creating optimised native machine

code at run-time. Although there is hidden Byte and native code compilation, the text-code dualism is preserved, and anytime data and functions can be serialised to text and deserialised to code, too. JavaScript can be executed on a wide range of devices and in a wide range of application programs including WEB browser making this language an ideal candidate for an agent behaviour programming language, addressing physical and computational agents.

1.10 Agent-based Computing with the JavaScript Agent Machine (JAM)

JAM is the JavaScript Agent Machine, implemented entirely in portable JavaScript. *JAM* executes reactive agents based on the Activity-Transition-Graph behaviour model (ATG), programmed in *AgentJS* (Sub-set of *JavaScript* with some minor changes).

JAM can be executed on any JS Virtual Machine (VM), e.g., *nodejs*, *spidermonkey* with WEB browsers, and the new low-resource machine *jvm* (based on Samsungs *jerryscript* and *Iot.js*). *Jvm* can be used on any host platform, including Android and iOS mobile devices or micro controller-based beacons (Raspeberry PI). Finally, *JAM* can be integrated in Cordova-based Apps for Android and IOS devices, e.g., used for and performing Crowd Sensing with participatory and ad-hoc opportunistic user interaction.

JAM agents are composed of activities performing actions (computation, interaction, mobility) and (conditional) transitions between activities based on the agent state. Activities represent short term goals (commitments and intentions in terms of the Belief-Desires-Intention (BDI) agent architecture). The behaviour of a reactive activity-based agent is characterised by an agent state, which is changed by activities. Activities perform perception, plan actions, and execute actions modifying the control and data state of the agent. The agent behaviour and the action on the environment is encapsulated in agent classes (see Def. 1), with activities representing the control state of the agent reasoning engine, and conditional transitions connecting and enabling activities. Activities provide procedural agent processing by sequential execution of data processing and control statements. Agents can be instantiated from a specific class at run-time.

```
1: // Agent Class Constructor
2: function ac (p1,p2,..) {
```

```
 3:    // 1. Body Variables
 4:    this.v1 = expression
 5:    this.v2 = expression
 6:    ..
 7:    // 2. Activities
 8:    this.act = {
 9:        a1 : function () { statements },
10:        a2 : function () { statements },
11:        ..
12:        an : function () { statements }
13:    }
14:    // 3. Transitions of Activities
15:    this.trans = {
16:        ai : aj,
17:        aj : function () { return ax },
18:        ..
19:    }
20:    // 4. Optional Signal Handlers
21:    this.on = {
22:        sig : function (arg,from) { statements },
23:        ..
24:    }
25:    // 5. The "Program Counter"
26:    this.next = ai
27: }
```

Def. 1. Agent behaviour class template and constructor function in AgentJS (*JavaScript*)

The activity-transition graph based agent model is attractive for fine-grained agent scheduling. An activity is always executed atomically, but after an activity terminates, it is a well defined break point for agent process scheduling. An activity is activated by a transition depending on the evaluation of (private) agent data (conditional transition) related to a part of the agents belief in terms of the BDI architecture, or using unconditional transitions (providing sequential composition). Each agent belongs to a specific parameterisable agent class AC, specifying local agent data (only visible for the agent itself), types, signals, activities, signal handlers, and transitions. In contrast to common JavaScript objects, an *AgentJS* class definition may not use any references to free variables or functions. The *this* variable references always the agent object, and can be used, e.g., in transition functions, handlers, activities, and first order functions directly.

Agent interaction in MAS is characterised by synchronisation and data exchange. *JAM* provides interaction via the tuple-space communication paradigm with a set of simple and coordinating access operations (input, output, read, remove, test), which is well accepted and an understood approach in distributed programming. Additionally, signals can be used for simple distributed one-way notifications carrying simple data. Tuple space communication is anonymous, generative, and data-centric, whereas signals are messages directed to specific (identified) agents.

The Agent Input-Output System (*AIOS*) is the *interface and abstraction layer* between agents programmed in *AgentJS* and the agent processing platform (*JAM*), shown in Fig. 10. Furthermore, it provides an interface between host applications and *JAM*, e.g., a Cordova-based Android or iOS App. The *AIOS* provides computational functions, code (ATG) modification, agent control (creation, forking, termination), agent mobility, agent interaction (tuple space, signals), and APIs for other module like Machine Learning (ML), SAT solver, and Constraint Programming. The *AIOS* is extensible, and additional module API can be added at run-time by the platform. An agent scheduler is used to execute multiple agents time multiplexed. Agent execution is encapsulated in a process container handled by the *AIOS*. An agent process container can be blocked waiting for an internal system-related IO event or suspended waiting for an agent-related *AIOS* event (caused by the agent, e.g., the availability of a tuple). Both cases stops the agent process execution until an event occurred. Finally, *AIOS* provides agent resource control.

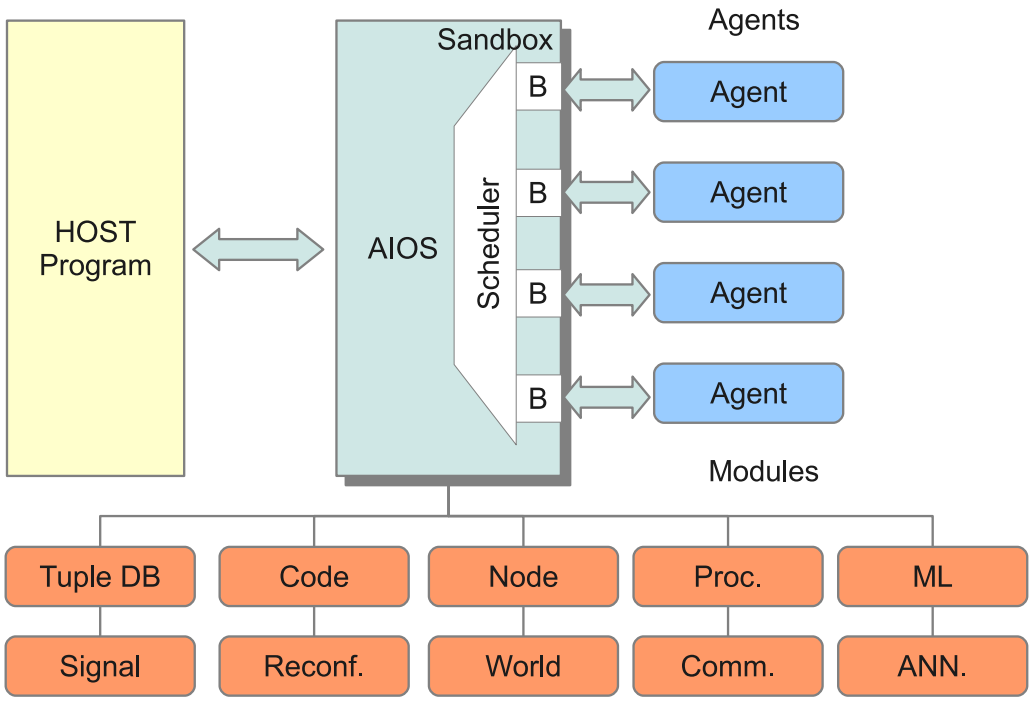

Fig. 10. Interface between agents and JAM and between JAM and a host application by the Agent Input-Output System (AIOS)

To distinguish at least trustful and not trustful agents, different agent privilege levels were introduces, providing different *AIOS* API sets, with level 0 as the lowest level that grants agents only computational statements. They can perform negotiation to get an higher level based on protected and secured capabilities. Level 1 agents can access the common *AIOS* API operations, including agent replication, creation, killing, sending of signals, and code morphing. Level 2 agents are capable to negotiate their desired resources on the current platform, i.e., CPU time and memory limits. Finally, level-3 agents are system agents with an extended API and full platform control, with unlimited resources, but they are stationary (immobile) and bound to the agent platform. Physical agents in simulations are always level-3 agents.

An agent of level *n* may only create agents up to level *n*. Level-3 agents can initially only be created by the *JAM* platform. After a migration the

destination node decides about the privilege level and can lower it, e.g., considering the agent source being not trustful. A migrated agent can get a higher privilege level by capability-based negotiation, requiring a valid platform capability with the appropriate rights.

1.10.1 Physical and logical nodes

A *JAM* instance is a physical agent processing node consisting of at least one logical (virtual) *JAM* node contained in a *JAM* world. Additional logical nodes (*vJAM*) can be created at run-time. Multiple logical nodes can be connected within this world by virtual links, a prerequisite for simulation. Logical nodes can be arranged in grid networks (one, two- or three-dimensional) or arbitrary graphs, e.g., virtual mobile ad-hoc networks. All logical nodes and agents are processed by the same physical instance. Actually, there is no benefit for parallel processing.

1.10.2 Connectivity and Communication

Physical *JAM* nodes can be connected via arbitrary communication links using the Agent Management Port (AMP) protocol. *AMP* supports agent process migration, signal propagation, and general node control. Supported *AMP* ports are:

- Stream Sockets
- File system Pipes
- UDP Sockets
- TCP Sockets
- HTTP Sockets
- Web Sockets
- RS232 (Serial interface)
- Bluetooth

Nodes are connected via loosely coupled links via negotiation and periodic check-pointing (ping-pong messages). Logical nodes are connected via virtual channel links (queues). Only logical nodes can connect to other logical nodes (either of the same physical node or of different physical nodes).

1.11 Agent-based Simulation and Computation Environment for JAM

The *JAM* agent platform was originally used for the ABC domain to implement distributed AI systems. Testing loosely coupled distributed systems is a challenge and the demand for simulation of computational agents deployed in virtual artificial worlds raised. Commonly, there is a large gap between real world data processing and simulation of data processing and communication. To overcome this gap, a simulation layer was added on the top of the *JAM* platform enabling the simulation of distributed data processing by computational *JAM* agents (i.e., ABCS). Finally, the simulation layer was extended by the concept of physical agents to enable simulation of physical entities closely coupled to computational agents (i.e., ABXS).

The *SEJAM* simulator is used to create a simulation world (consisting of individuals represented by agents) that is attached to the Internet enabling remote Crowd Sensing with mobile computational agents. The *SEJAM* simulator is basically a graphical user interface (GUI) on the top of a *JAM* node, shown in Fig. 11.

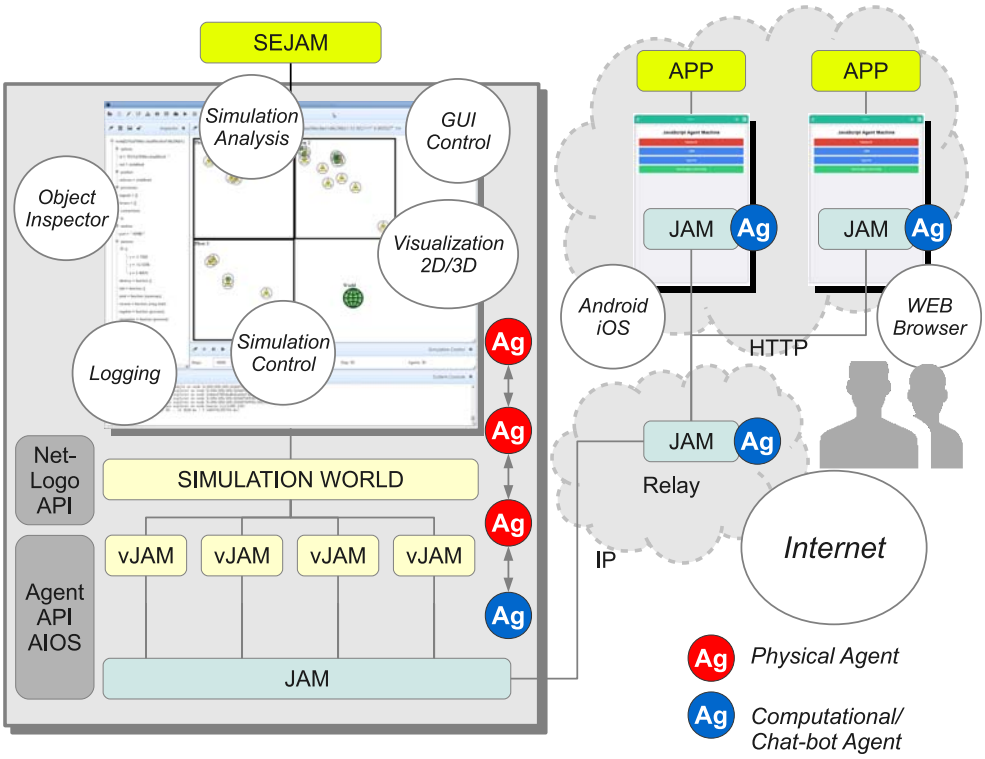

Fig. 11. Principle concept of closed-loop simulation for augmented virtuality: (Left) Simulation framework based on the JAM platform (Right) Mobile and non-mobile devices executing the JAM platform connected with the virtual simulation world (via the Internet) (Bosse et al., 2018)

The simulation world consists of multiple virtual *JAM* nodes controlled by the one physical node. Any logical node can connect to remote nodes via the Internet, e.g., using HTTP or UDP communication.

The entire simulation model is specified in a *JavaScript/JSON* objects containing:

- Agent behaviour code and visual properties (distinguishing physical and computational agents);
- Node visual properties (representing computers or physical entites);
- Communication links and resources;

- Simulation parameter;
- World description (spatial structure).

SEJAM supports programming and processing of physical and computational agents using the same *AgentJS* programming model. In *SEJAM*, an agent is specified by its behaviour code and visual properties (for visualisation). A physical agent can access a dedicated API only available in the simulator supporting *NetLogo* statements *create*, *ask*, and *set*. Furthermore, a two-dimensional patch-grid world can be created associating variables to patches at specific positions. Creation of a physical agent always creates always a virtual JAM node. too. Modification of the position of a physical agent within the simulation world moves the node together with the agent.

1.11.1 JAM Check-pointing

Check-pointing creates a snapshot of the entire state of a process. A check-point of a *JAM* instance can be created for later or parallel processing by another *JAM* instance. The snapshot contains:

- All agent processes including data state and control structures;
- The entire tuple space data base;
- The world configuration and contained all nodes;
- The world and node states, the control structures including pending signals;
- All agent class templates.

A forked *JAM* instance snapshot can be processed by another simulator (including additional visual states) to enable co-simulation from an arbitrary simulation time point under different conditions, e.g., using Monte Carlo simulation, or on different time scales. The latter allows co-simulation to get future development of simulation snapshots that can be back-propagated in the parent simulation to influence the propagation of the simulation worlds with future knowledge. This is an important feature if the simulation is influenced continuously by the real world and the real world is influenced by the simulation.

1.12 Time Machine: Multi-time scale and multi-trace Simulation

Coupling virtual worlds in simulation with real worlds in real-time enables

real-world synchronised simulation, assuming that the virtual world is related to a spatial area of the real world. The aforementioned capability of the *JAM* platform and the *SEJAM* simulator to create simulation snapshots enables the forking of simulation snapshots to predict future developments of the real world state.

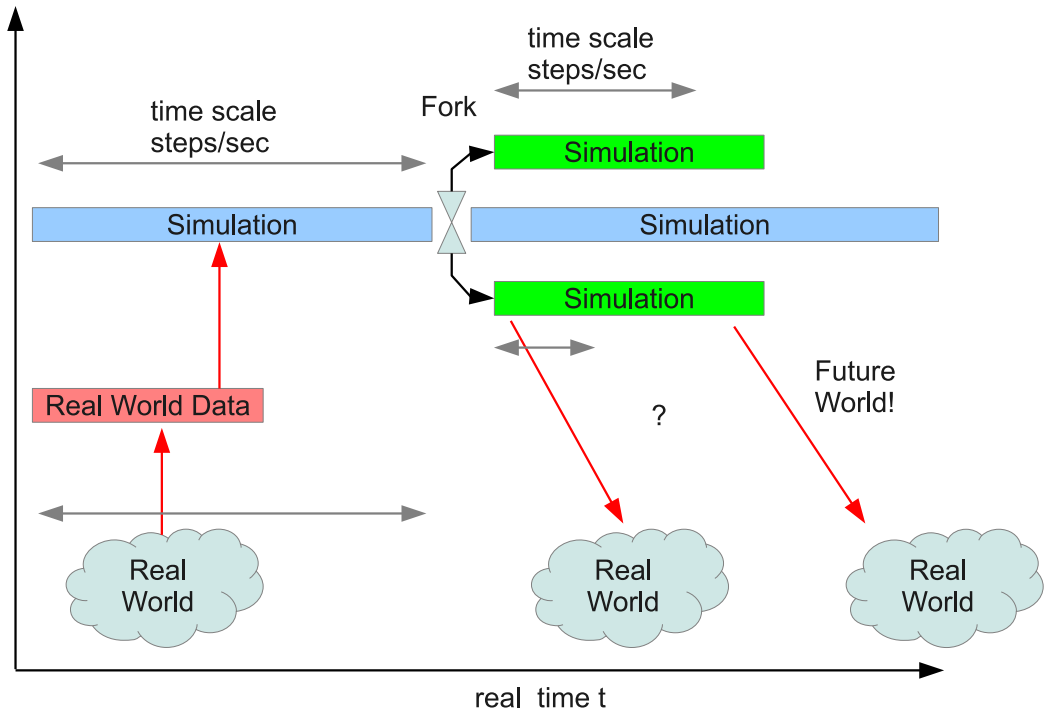

Fig. 12. A time machine using forked simulation snapshots to predict real world development in the future

1.13 A Technical Use Case: Environmental Control

A simple proof-of-concept study demonstrates the capabilities of the augmented virtual simulation world for the control of technical infrastructure systems like environmental illumination.

The primary goal of this demonstrator is the control of city environments by the combination of crowd sensing with simulation to optimise environmental situations, e.g., to save energy or to increase people's well being and satisfaction with domestic services. The simple demonstrator (Bosse et

al., 2018) consists of an artificial street area with moving nodes (representing humans interacting with mobile devices or machines), beacons (access points) for sensor aggregation and distribution, and some external beacons connected to the Internet enabling the connection to mobile devices via the Internet, and finally smart light devices illuminating streets and buildings based on feedback from people and sensors, shown in Fig. 13.

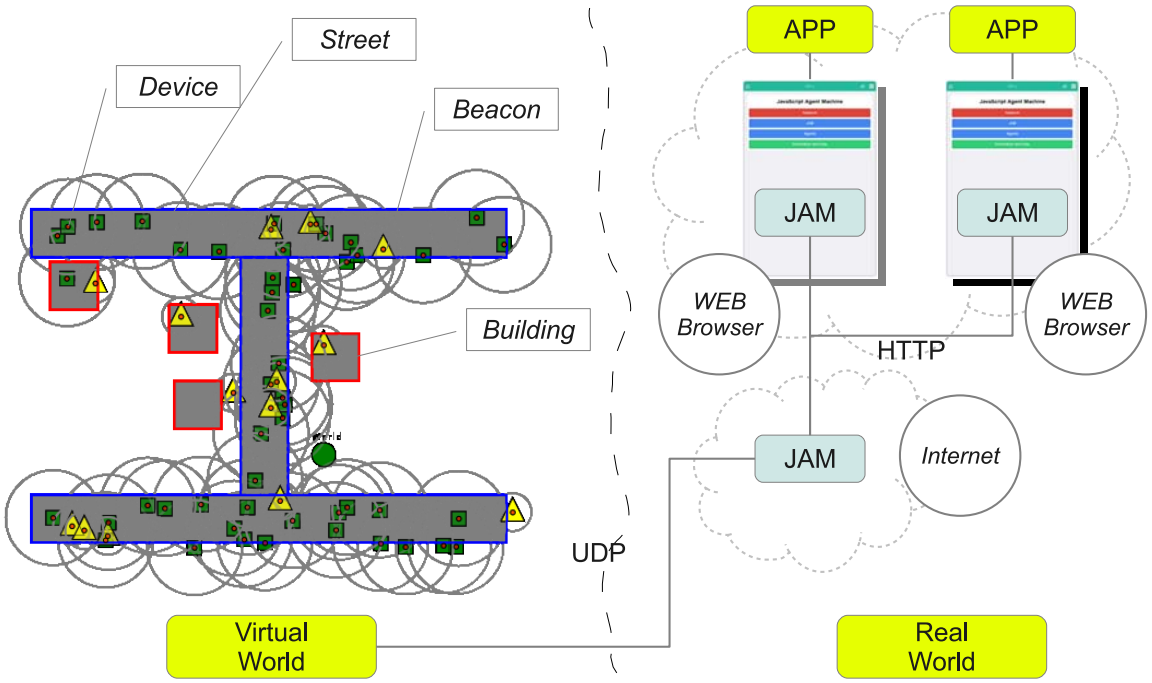

Fig. 13. A demonstrator: (Left) Simulation with artificial street areas consisting of street segments (blue rectangle) and buildings (red rectangles), beacons (yellow triangles, vJAM node), populated with stationary and mobile devices (green squares, vJAM node). The grey circles around beacons and nodes show the wireless communication range. Only overlapping circles connect nodes. (Right) Crowd Sensing system with mobile Apps connected to the simulation via the Internet (Bosse et al., 2019)

The goal of explorer agents (sent out by mobile or light devices) is information mining in the outside world via Internet deployed agent processing

platforms that can be accessed by simulation nodes and WEB browser Apps. The collected information is passed back to the root node (e.g., a mobile device of a passenger) to assist decision making and navigation.

The explorer agents have to estimate the position of the root node by performing sensor mining from surrounding devices they are visiting and from the outside world (far away) by asking questions answered by humans via the WEB App chat dialogue. The explorer asks a user for its current place and location within a region of interest and an assessment of the current light situation. Depending on the answer a specific action is suggested.

Based on answered questions regarding the current user location, the satisfaction of ambient light condition, and an optional fusion with device sensor data (light, position , and so on), actions are directed to smart light control devices to change the light condition in streets and buildings by using mobile notification agents. The action planning bases on the Crowd demands and energy saving constraints. If action is required, mobile notification agents are sent out to neighbouring nodes to change light intensity based on directed diffusion, random walk, and divide-and-conquer approaches.

The simulation with the current *SEJAM* simulator was able to be performed with more than 1000 nodes, hundreds of beacons, and more than 10000 explorer agents. Real time values of one simulation step depends on the number of active agents to be processed, node and agent mobility (graphics and communication), and ranges from 1 ms to 1 s.

1.14 A Simulation Use Case: Sakoda Segregation and Parameter Variance by Digital Twins

The previous demonstrator use case was primarily used for the control of technical real world environments by a combination of simulation and Crowd Sensing. The simulation results can be directly used to modify the real world environment. The following demonstrator is primarily used to study social and socio-technical systems by ABM and ABS, combined with Crowd Sensing basically to get input data for the simulation.

The MAS provides the bidirectional integration of DM via Crowd Sensing and ABMS (i.e. applying DM in ABMS by creating generated simulation data and applying ABMS in DM by back propagating generated simulation data). The framework is suitable to combine social and computational simulations with real-world interaction at run-time by integrating Crowd

Sensing, using mobile computational agents interacting with physical agents.

In this demonstrator, a simple social interaction model between different groups is assumed (based on Sakoda (Medina et al., 2017)). The social model expresses the social expectation at a specific location in dependence on the neighbouring population. A set of parameters $\boldsymbol{S}=\{s_{i,j}\}$ expresses the social relation between individuals of the same group and between different groups. There are three simple states: $s_{i,j}=\{-1,0,1\}$ expressing group proximity (-1: disgusting, 0:neutral, 1:attractive), with one $s_{i,j}$ parameter for each group combination i and j. The social expectation at a specific location influences migration mobility of individuals to improve their social expectation. The parameter set $\boldsymbol{S}$ can be pre-selected and unique for all individuals or derived from Crowd Sensing via a chat bot addressing the digital twin concept (see Fig. 14).

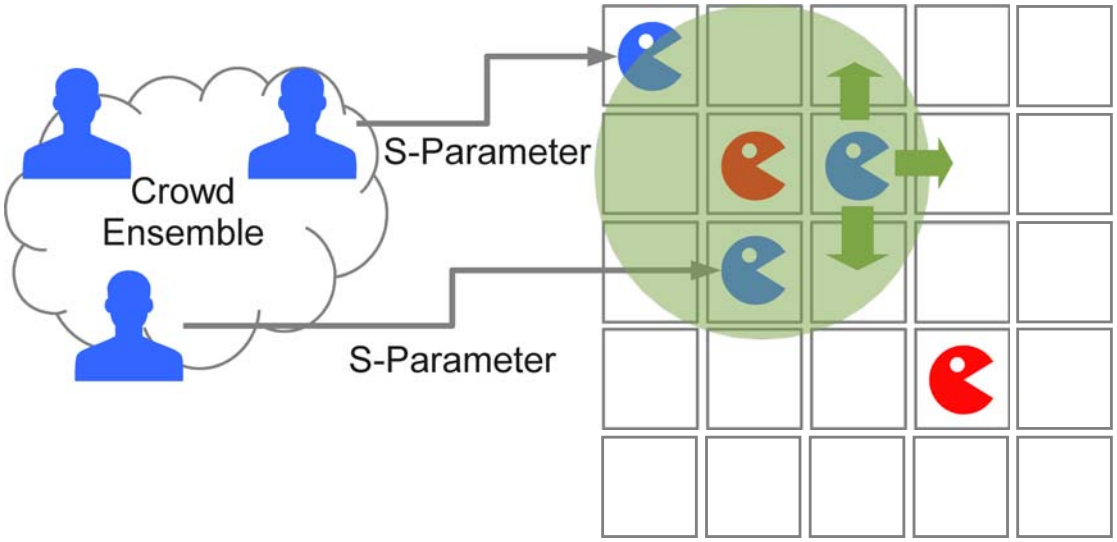

Fig. 14. Digital twins created from Crowd Sensing by paramterising the Sakoda social interaction model influencing the migration mobility of individuals to improve their social expectation

The Sakoda behaviour model can be applied on two different spatial and time scales resulting in different self-organising behaviour:

1. Long-time and long-range scale of mobility addressing classical segregation, e.g., in cities and countries, and
2. Short-time and short-range scale of mobility addressing, e.g., city mobility.

1.14.1 Interaction Model

Assuming a two-dimensional grid world that consists of places at discrete locations (x,y), an artificial agent can occupy one place of the grid. Maximal one agent can occupy a place. The agents can move on the grid and can change their living position. It is assumed that there are two groups related to the classes a and b of individuals. The social interaction is characterised by different attitudes (Medina et al., 2017) of an individual between different and among same groups given by four parameters:

$$S = (s_{aa}, s_{ab}, s_{ba}, s_{bb}) \tag{2}$$

The model is not limited to two groups of individuals. The S vector can be extended to four groups (or generalised) by the matrix:

$$S_{abcd} = \begin{pmatrix} s_{aa} & s_{ab} & s_{ac} & s_{ad} \\ s_{ba} & s_{bb} & s_{bc} & s_{bd} \\ s_{ca} & s_{cb} & s_{cc} & s_{cd} \\ s_{da} & s_{db} & s_{dc} & s_{dd} \end{pmatrix} \tag{3}$$

The world model consists of N places x_i. Each place can be occupied by none or one agent either of group α or β, expressed by the variable x_i={0,-1,1}, or generalised x_i={0,1,2,3,4,..,n} with n groups. The social expectation of an individual i at place x_i is given by:

$$f_i(x_i) = \sum_{k=1}^{N} J_{ik}\delta_s(x_i, x_k) \tag{4}$$

The parameter J_ik is a measure of the social distance (equal one for Moore neighbourhood with distance one), decreasing for longer distances. The parameter δ expresses the attitude to a neighbouring place, given by

(for the general case of *n* different groups):

$$\delta_s(x_i, x_k) = \begin{cases} s_{\alpha\beta} & , \text{ if } x_i \neq 0 \text{ and } x_k \neq 0 \text{ with } \alpha = x_i, \beta = x_k \\ 0 & , \text{ otherwise} \end{cases} \quad (5)$$

Self-evaluation is prevented by omitting the current place (i.e., $k \neq i$). There is a mobility factor *m* giving the probability for a movement.

An individual agent ag_i of any group α (class from the set of groups) is able to change its position by migrating from an actual place x_i to another place x_m if this place is not occupied (x_m=0) and if $f_i(x_m) > f_i(x_i)$ and the current mobility factor m_i is greater than 0.5. Among social expectation (resulting in segregation), transport and traffic mobility has to be considered by a second goal driven function $g(x_m)$, commonly consisting of a destination potential functions with constraints (e.g., streets). If $g_i(x_m) > f_i(x_m) > f_i(x_i)$ than the goal driven mobility is chosen, otherwise the social driven.

$$g : (x_i, x_m, x_k, v, t) \rightarrow \mathbb{R} \quad (6)$$

The mobility function *g* returns a real value [0,1] that gives the probability (utility measure) to move from the current place x_i to a neighbouring place x_m to reach the destination place x_k with a given velocity *v*. The *g* function records the history of movement. Far distances from the destination increases *g* values. Longer stays at the same place will increase the *g* level with time *t*. Social binding (i.e., group formation) will be preferred over goal driven mobility.

The computation of the neighbouring social expectation values *f* is opportunistic, i.e., if *f* is computed for a neighbouring node assuming the occupation of this neighbouring place by the agent if the place is free, and the current original place is omitted x_i for this computation. Any other already occupied places are kept unchanged for the computation of a particular *f* value. From the set of neighbouring places and their particular social and mobility expectations for the specific agent the best place is chosen for migration (if there is a better place than the current with the above condition). In this work, spatial social distances in the range of 1-30 place units are considered.

Originally, the entire world consists of individual agents interacting in the world based on one specific set of attitude parameters *S*. In this work the

model is generalised by assigning individual entities its own set *S* retrieved from real humans by crowd sensing, or at least different configurations of the *S* vector classifying social behaviour among the groups. Furthermore, the set of entities can be extended by humans and bots (intelligent machines) belonging to a group class, too.

Segregation effects inhibit individual movement until a different social situation enables a movement. Transportation mobility triggers movement even if there is no social enabler. This reflected in the extended Sakoda model by the mobility factor *m* and the goal driven expectation function *g* that control mobility and overlays social and transportation and traffic mobility. The mobility function *g* includes random walk and diffusion behaviour, too. Constrained mobility is one major extension of the original Sakoda model presented in this work.

1.15 Model Parameters and Crowd Sensing

Creation of virtual digital twins is the aim of the crowd sensing task. The crowd sensing is performed with chat bot agents. One stationary agent is operating on a user device (chat moderator agent), e.g., a smart phone, and another mobile agent is responsible to perform the survey (either participatory with a former negotiation or opportunistic ad-hoc) by providing a dialogue script consisting of questions and text. The results of the survey, a set of answered questions, are used to derive the following simulation model parameters, shown in Def. 2.

```
parameters = {
    group : string "a"|"b"|..,
    social-distance: number [1-100],
    mobility-distance: number [1-100],
    social-attitudes : [saa,sab,sba,sbb] | number [][],
    mobility : number [0-1],
    position : {x:number,y:number},
    destination : {x:number,y:number}
}
```

Definition 2. Sakoda model parameters

1.15.1 Survey Questionnaire

The following Tab. 1 summarizes the ad-hoc mobile survey performed for twin creation by mobile chat-bot agents created in the simulation world.

There are questions asking directly for the user choice of their class (poor and rich, related to agent classes A and B, respectively), which can correct or incorrect (i.e., the user decision is highly subjective). And there are question trying to estimate a more accurate class membership. Finally, some questions measure the social interaction radius and the mobility probability.

Qid / Variable	**Type, Value, Condition**	**Text**	**Input**
M1	Message	*Welcome..*	-
QSN / S	Sensors	-	{DATE, BROWSER, OS, GPS, GPS5, LOCATION, LOCATION5}
QCL / C	Question, Mutual Choice	*Who are you? Do you think you are poor or rich?*	{Poor, Rich, Do not know}
QR1 / aP	Question, Value choice, $1=QCL.answer if QCL.answer≠"Do not know"	*How do you rate your relation to people of your own group $1? Do you like (1) or dislike (-1) them?*	{-1, 0, 1}
QR2 / aN	Question, Value choice, $1=¬QCL.answer if QCL.answer≠"Do not know"	*How do you rate your relation to people of the other group $1? Do you like (1) or dislike (-1) them?*	{-1, 0, 1}
QFN / fN	Question, $1=¬QCL.answer if QCL.answer ≠ "Do not know"	*Do you have $1 friends?*	{ yes, no }
QA / aG	Question, Value	*How old are you?*	[1,100]
QMS / iC	Question, Mutual Choice	*What is your monthly salary?*	{<1000, 1000-3000, 3000-5000, >5000}
QT / sA	Question, Multiple Choice	*What are your spare time activities?*	{Travelling, Sports, Education/Art, Social, Friends, Gardening, TV/Streaming}

Qid / Variable	Type, Value, Condition	Text	Input
QL / aR	Question, Text, Default=QSN.LOCATION5.CITY or QSN.LOCATION.CITY	*Where do you live [Enter ZIP Code or City Name]?*	ZIP / City
QSC / nS	Question, Value	*How many social contacts do you have?*	[1,100]
QMD / mM	Question, Mutual Choice	*Do you like to move?*	{Yes, Maybe, No}
M2	Message	*Thank you!*	-

Tab. 1. Questions and Variables of the survey dialogue

1.15.2 Model Parameter Estimation

The model parameters of the digital twins are derived from the survey results.

Coding of categorical parameters

$$\phi(C) = \begin{cases} 0, C = \text{Do not know} \\ 1, C = \text{Poor} \\ 2, C = \text{Rich} \end{cases}, \phi(iC) = \begin{cases} 1, iC = <1000 \\ 2, iC = 1000\text{-}3000 \\ 3, iC = 3000\text{-}5000 \\ 4, iC = >5000 \end{cases}$$

$$\phi(sA) = \sum \begin{cases} 1, sA = \text{TV/Streaming} \\ 2, sA = \text{Friends} \\ 4, sA = \text{Social} \\ 8, sA = \text{Sports} \\ 16, sA = \text{Gardening} \\ 32, sA = \text{Education/Art} \\ 64, sA = \text{Travelling} \end{cases}, \phi(mM) = \begin{cases} -1, C = \text{No} \\ 0, C = \text{Maybe} \\ 1, C = \text{Yes} \end{cases} \quad (7)$$

Estimating digital twin class

- The following class estimation (*a*/*b*) bases not only on the user class selection. It uses additional survey variables

$$class = \text{Prio} \begin{cases} a, \phi(C) = 0 \wedge \phi(iC) = 1 \\ a, \phi(C) = 0 \wedge \phi(sA) < 16 \\ b, \phi(C) = 0 \wedge \phi(iC) > 3 \\ b, \phi(C) = 0 \wedge \phi(sA) > 30 \\ a, \phi(C) = 1 \wedge \phi(iC) = 1 \wedge \phi(sA) < 16 \\ b, \phi(C) = 1 \wedge (\phi(iC) > 2 \vee \phi(sA) > 90) \\ b, \phi(C) = 2 \wedge \phi(iC) > 2 \wedge \phi(sA) > 90 \\ a, \text{otherwise} \end{cases} \tag{8}$$

Estimating digital twin model parameters

$$\begin{aligned} s_{aa} &= \begin{cases} aP, class = a \\ 0, class = b \end{cases}, s_{bb} = \begin{cases} 0, class = a \\ aP, class = b \end{cases} \\ s_{ab} &= \begin{cases} aN, class = a \\ 0, class = b \end{cases}, s_{ba} = \begin{cases} 0, class = a \\ aN, class = b \end{cases} \\ s_{mobi} &= \begin{cases} 1, \phi(mM) = 1 \\ 0.5, \phi(mM) = 0 \\ 0.2, \phi(mM) = -1 \end{cases}, s_{sodist} = \begin{cases} 2, nS < 10 \\ 3, nS < 30 \\ 5, nS \geqslant 30 \end{cases}, \end{aligned} \tag{9}$$

with Φ(x) giving the numerical code of a categorical variable (multiple choices answers are summed by a weighted code).

An ad-hoc survey with 20 replies returned by users was performed via the WEB *JAM* App connected to a public *JAM* relay within a time interval of two hours. The survey is not representative and was used only for a proof of concept. The statistical analysis results (wtih coded categorical variables) are shown in Fig. 15.

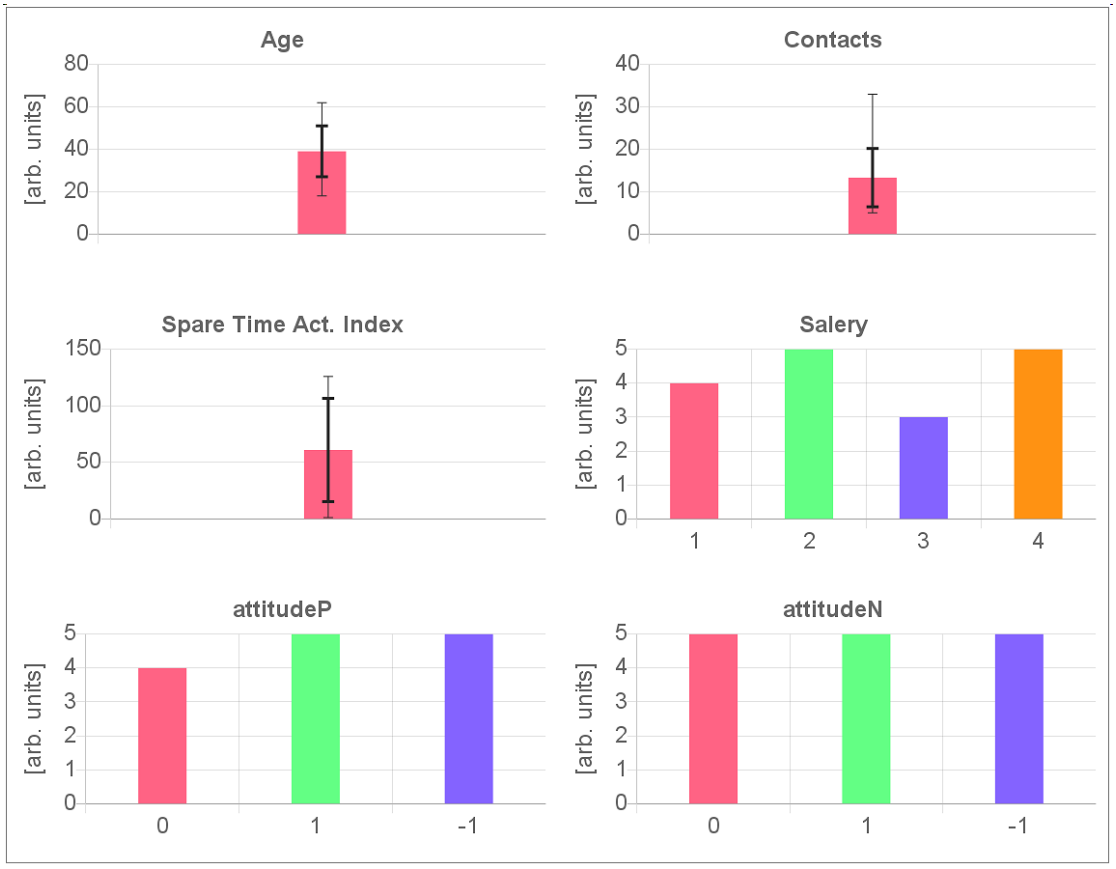

Fig. 15. Statistics of some of the variables derived from the ad-hoc mobile survey using the JAM App. (Mean, standard deviation, and minimal and maximal values are plotted for the numerical variables)

Fig. 16 shows the class distribution from the survey (user selection) and the class estimation by the twin model shown above. Surprisingly the majority shifts from rich to poor classification!

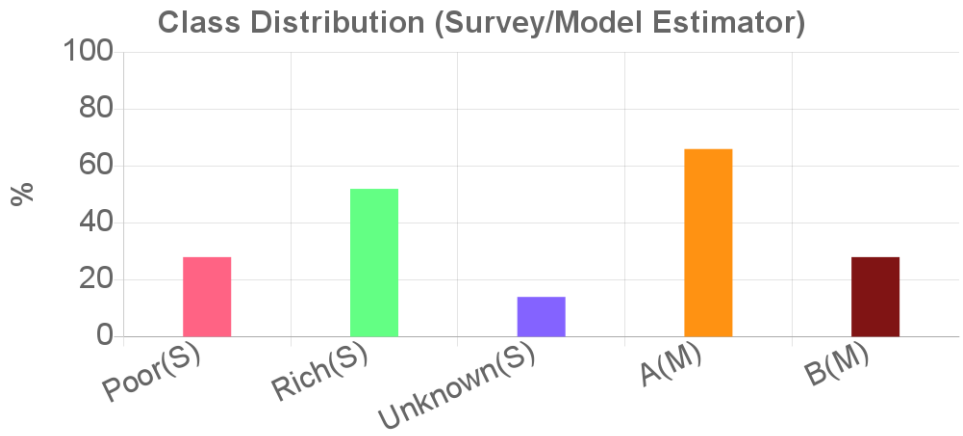

Fig. 16. Comparison of survey classification by users (Poor class relates to A, rich to *B*, respectively) and estimated twin class (A/B).

The injection of digital twins in the simulation at run-time leads to the following observations and influences on the outcome (emergence) of the entire multi-agent system:

1. A mixed and heterogeneous agent distribution is emerging by adding variance of agent behaviour to the system, i.e., there is a large set of different agent behaviour (originally there set of agent behaviour only consisted of two different parameter sets).
2. The dynamics of the segregation and group formation (creating local clusters) is strongly increased.
3. There is more agent mobility and fluctuation.
4. The time scale of non-stability (agent mobility) is extended by two or three times.
5. There are inversions of the *a*/*b* ratio of mean ensemble social expectation and suddenly appearing steps of change of the mean social expectation for one or both groups.

One important emergent measure variable is the globally averaged social expectation value (i.e., the social satisfaction measure) of agents at their current place. Surprisingly, different simulations showed totally different

global satisfaction developments for class *A* and class *B* agents. Some examples are shown in Fig. 17. Sometimes, there is a higher global social satisfaction for one class of about 40%, sometimes they are equal! It seems that the initial spatial distribution of agents determines the final stationary state of the society (agents are placed at random locations with a randomly chosen class using Monte Carlo simulation).

The influence of digital twins injected in the simulation on the development of the social expectation measure is significant, as shown in Fig. 19. The digital twins (200 overall) are injected in the first 100 simulation time units. The influence on the global dynamic can occur more than 200 simulations after their injection (e.g., shown in Fig. 19 (d)). Furthermore, the digital twins with a wide range of model parameter variance can trigger steps in the social expectation measure (e.g., in Fig. 19 (a) at 100 and 150 time units). The averaged social expectation measure of *A*/*B* agents is higher than in the experiments without digital twins (this relates clearly to a higher density of agents). But the social expectation measure of the digital twins is always significantly lower than *A*/*B* satisfaction (a result from the broad diversity and only loosely integration in clusters). Due to the low number of returned survey results 10 digital twins were created from one parameter set derived from a survey (and placed randomly on the living areas).

Finally, Fig 20 shows examples of the start and end situation of the simulation world with and without digital twins. Digital twins lead to the formation of larger group clusters, although due to broad model variance the digital twins are more spatially scattered than pure class *A*/*B* agents. The movement of agents is constrained. The simulation world is partitioned into living areas (gray areas). Agents can stay and move only in these areas.

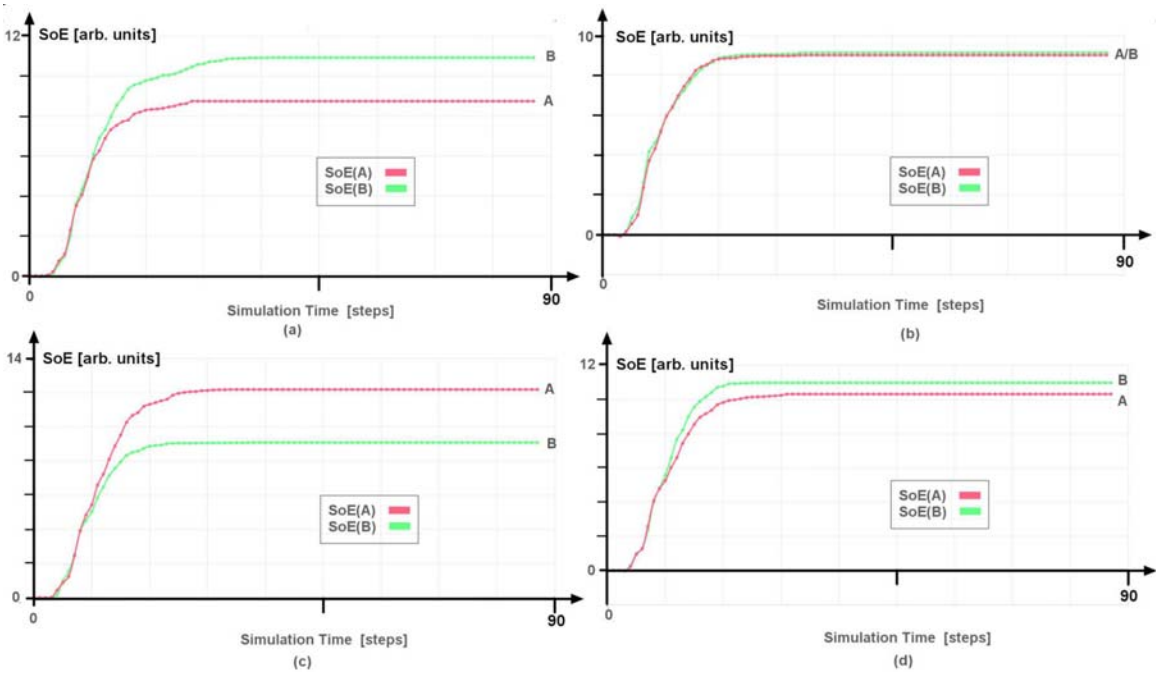

Fig. 17. Examples of different simulation outcomes of the globally averaged social expectation value (computed by Eq. 4) for 200 agents of class A (red line) and 200 agents B (green line) without digital twins.

The global averaged social expectation value (computed by Eq. 4) increases significantly in the presence of the additional digital twins, shown in Fig. 18. Additionally, the variance of the end stationary state of the multi-agent system is about 3 times higher than without digital twins.

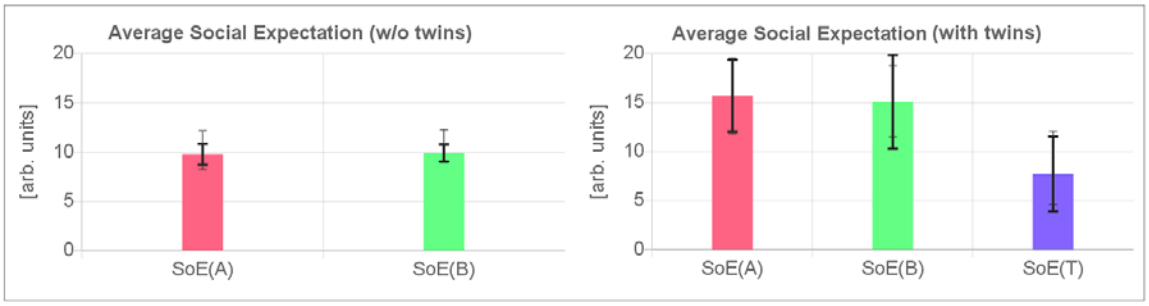

Fig. 18. Globally averaged social expectation values for an ensemble of simulation runs with and without digital twins

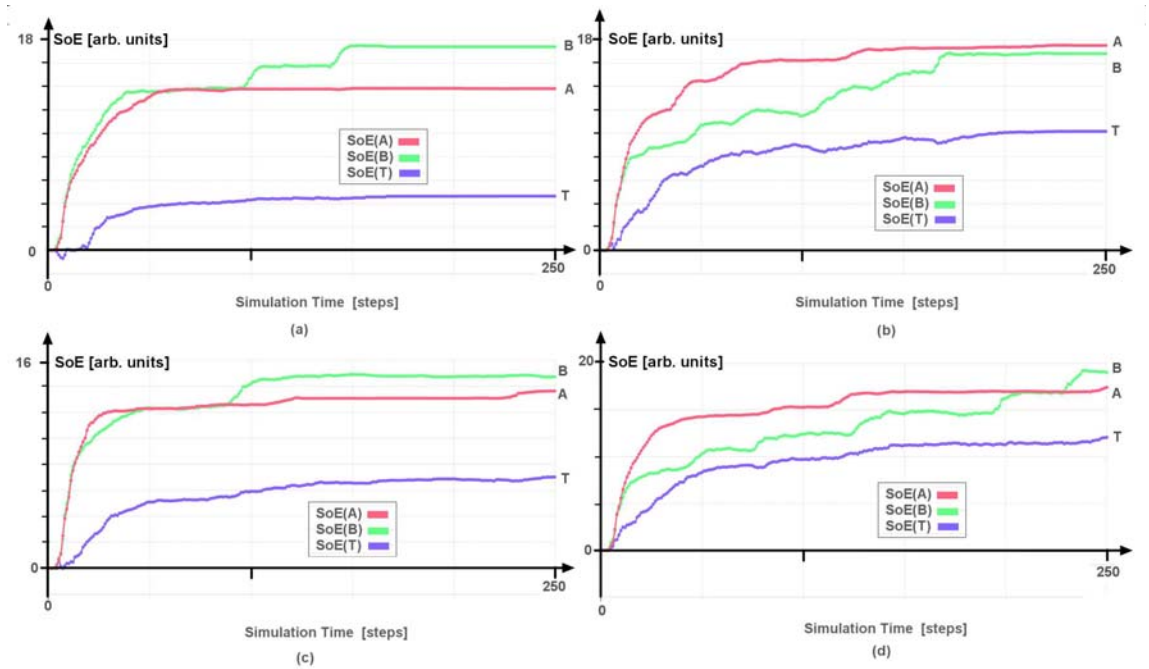

Fig. 19. Examples of different simulation outcomes of the globally averaged social expectation value (computed by Eq. 4) for 200 agents of class A (red line) and 200 agents B (green line) with additional 200 digital twins.

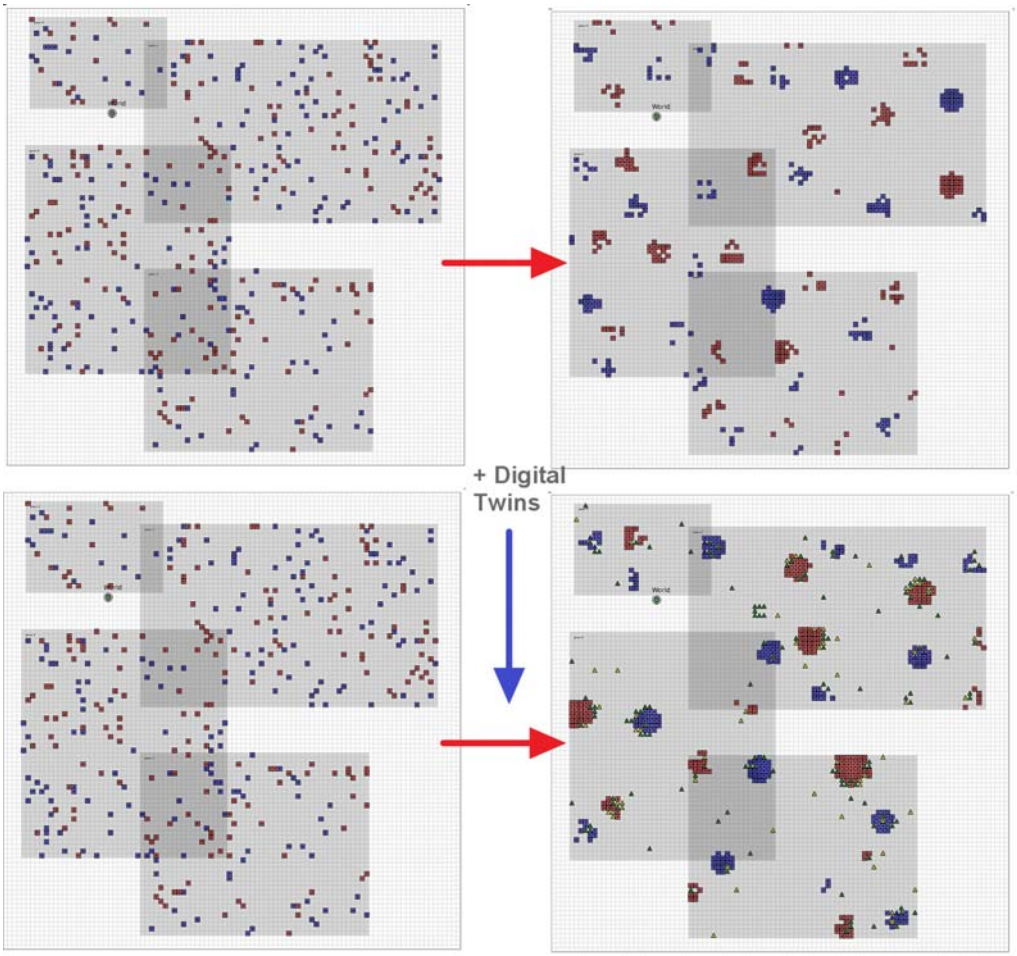

Fig. 20. Examples of different simulation world developments without (top) and with (bottom) digital twins (triangles).

1.16 Interactive Simulation: An Educational Tool

The previous section showed a proof-of-concept field study for augmented and real-time capable agent-based simulation. The real-time capability was in this case study not fundamental. There are other use cases (like traffic and crowd flow management) which rely significantly on the real-time capability.

Research and environmental control is only one relevant field of application for the augmented simulation approach. Education and studying human-machine interaction is another field that can profit from the

features of the introduced framework.

Combining the concepts of mobile crowd sensing, agent-based simulation, and digital twin mapping of humans interacting in real-time with simulation worlds can be a powerful educational tool. The aforementioned simulation of networking and segregation using digital twins derived from surveys as a pattern disturbance can be studied by students in social sciences interactively.

1.17 Summary

A novel approach was presented that provides an augmentation of virtual simulation worlds by using mobile computational agents and crowd sensing.

Mobile agents can be used to fusion real and virtual worlds composing augmented virtuality. The deployment of the *JAM* agent processing platform enables large-scale simulation of extended and complex socio-technical systems, e.g., complex traffic situations and smart wide-range traffic management, large-scale social media interaction, crowd sensing and interaction, and many more complex systems with unknown emergence behaviour.

There are basically two agent model classes covered by one unified agent model: Physical and computational agents, both relying on the same unified agent behaviour programmed in *JavaScript*.

Physical *JAM* agents relate to the ABM/ABS domains, whereas the computational agents relate to the ABC domain. Computational agents can migrate between real and virtual worlds seamlessly. Moreover, from the point of view of social science, decision making, opinion formation, and voting processes can be studied with a unified simulation and data mining framework.

A simple proof-of-concept study demonstrated the capabilities of the augmented virtual simulation wolrd for the control of technical infrastructure systems like environmental illumination.

An extended field study showed the influence of digital twins created from mobile ad-hoc surveys on segregation patterns and dynamic. An originally homogeneous two-class multi-agent system was enriched with model variance by users in the real world.

The tools (the big machine) are available and the suitability could be shown with demonstrators. Future work has to find suitable research ques-

tions that can be investigated with the machine from social sciences, smart city and facility management, production and logistics environments, warehouse systems, and many more fields of applications. Additionally, real-time issues by agent interaction in real and virtual worlds differing in real time scale have to be investigated.

One major and fundamental issue arising generally is the fusion of real and virtual worlds with respect to the temporal and spatial domains, which must be addressed in future research work. Useful temporal and spatial mappings has to be identified.

1.18 References

Al-Zinati, M., Wenkstern, R. (2017). *An agent-based self-organizing traffic environment for urban evacuations*, Proceedings of the The Sixteenth International Conference on Autonomous Agent and Multiagent Systems, AAMAS 2017, Sao Paulo, Brazil, May 2017.

Baqueiro, O., Wang, Y. J., McBurney, P., Coenen, F. (2008). *Integrating Data Mining and Agent Based Modeling and Simulation*, in *ICDM 2009, LNAI 5633*

Bosse, S. (2016). *Mobile Multi-Agent Systems for the Internet-of-Things and Clouds using the JavaScript Agent Machine Platform and Machine Learning as a Service*, in The IEEE 4th International Conference on Future Internet of Things and Cloud , 22-24 August 2016, Vienna, Austria, 2016, DOI: 10.1109/FiCloud.2016.43

Bosse, S. (2018). *Smart Micro-scale Energy Management and Energy Distribution in Decentralized Self-Powered Networks Using Multi-Agent Systems*, FedCSIS Conference, 6th International Workshop on Smart Energy Networks & Multi-Agent Systems, 9-12.9.2018, Posznan, Poland, 2018

Bosse, S., Engel, U. (2018). *Augmented Virtual Reality: Combining Crowd Sensing and Social Data Mining with Large-Scale Simulation Using Mobile Agents for Future Smart Cities*, Proceedings, Volume 4, ECSA-5 5th International Electronic Conference on Sensors and Applications 15–30 November, 2018 DOI 10.3390/ecsa-5-05762

Bosse, S., Pournaras, E. (2017). *An Ubiquitous Multi-Agent Mobile Platform for Distributed Crowd Sensing and Social Mining*, FiCloud 2017: The 5th International Conference on Future Internet of Things and Cloud, Aug 21, 2017 - Aug 23, 2017, Prague, Czech Republic

Bosse, S., Engel, U. (2019). *Real-time Human-in-the-loop Simulation with Mobile Agents, Chat Bots, and Crowd Sensing for Smart Cities*, Sensors (MDPI), doi: 10.3390/s19204356

Calenda, T. Benedetti, M. D., Messina, F., Pappalardo, G., Santoro, C. (2016). *AgentSimJS: A Web-based Multi-agent simulator with 3D capabilities*

Ganti, R. K., Ye, F., Lei., H. (2011). *Mobile Crowd Sensing: Current State and Future Challenges*. IEEE Communications Magazine, vol. 49, no. 11

Gilbert, N. (2004). *Agent-based social simulation: dealing with complexity*

Hamidi, H., Kamankesh, A. (2018). *An Approach to Intelligent Traffic Management System Using a Multi-agent System*, Int. J. ITS Res. , vol. 16, pp. 112-124

Hox, J. J. (2017). *Computational Social Science Methodology, Anyone?* Methodology, 13 (Supplement): 3 – 12. DOI: 10.1027/1614-2241/a000127

Leppäne, T. Lacasia, J. Á., Tobe, Y., Sezaki, K., Riekki. J. (2017). *Mobile Crowd Sensing with mobile agents*. Autonomous Agents and Multi-Agent Systems, vol. 31, no. 1, pp. 1-35

Mahrt, M., Scharkow, M. (2013). *The Value of Big Data in Digital Media Research*. Journal of Broadcasting & Electronic Media 57(1): 20-33, DOI: 10.1080/08838151.2012.761700

McFarland, D. A., Moody, J., Diehl, D., Smith, J. A., Thomas, R. J. (2014). *Network Ecology and Adolescent Social Structure*, American Sociological Review, http://dx.doi.org/10.1177/0003122414554001

McGrath, M. J., Scanaill, C. N. (2014). *Sensor Technologies Healthcare, Wellness, and Environmental Applications*, Apress Open, 2014.

Medina, P., Goles, E., Zarama, R., Rica, S. (2017). *Self-Organized Societies: On the Sakoda Model of Social Interactions*, Complexity

Pournaras, E., Moise, I.,Helbing, D. (2015). *Privacy-preserving Ubiquitous Social Mining via Modular and Compositional Virtual Sensors*, in IEEE 29th International Conference on Advanced Information Networking and Applications

Shah, D. V., Capella, J. N., Neuman, W. R. (2015). *Big Data, Digital Media, and Computational Social Science: Possibilities and Perils*. The

ANNALS of the American Academy of Political and Social Science 659: 6 – 13, DOI: 10.1177/000271621557208

Wang, F.-Y. (2005). *Agent-Based Control for Networked Traffic Management Systems*, Intelligent Transportation Systems, no. September/October,

Wilensky, U., Rand, W. (2015). *An Introduction to Agent-Based Modeling Modeling Natural, Social, and Engineered Complex Systems with NetLogo*,. MIT Press.

van Dam, K. H., Lukszo, I., Zofia. N. (2013). Eds., *Agent-Based Modelling of Socio-Technical Systems*. Springer Berlin