A Model and Platform for Building Agent-Based Pervasive Mixed Reality Systems

Angelo Croatti(✉) and Alessandro Ricci

Computer Science and Engineering Department (DISI),
Alma Mater Studiorum – University of Bologna, Via Sacchi, 3, Cesena, Italy
{a.croatti,a.ricci}@unibo.it

Abstract. In agent literature, a partially unexplored area is related to the integration of ever-wider opportunities offered by technologies such as Mixed Reality (MR) and Augmented Reality (AR). In this paper we present a framework called *Augmented Worlds* (AW), which provides a model and a technological support to develop a broad spectrum of agent-based AR/MR systems. Distinguishing key features of the approach include: bi-directional augmentation, support for existing cognitive agent technologies, support for developing open multi-user environments. In the paper, we describe first the conceptual model on which the framework is based, and then a concrete architecture and prototype implementation. Two case studies about real-world applications – an augmented museum and an augmented harbour – engineered with the framework are finally discussed.

Keywords: Mixed Reality · Augmented Reality
Augmented Worlds · Agent programming technologies · BDI

1 Introduction

AR and MR are quickly becoming mainstream technologies to be exploited for designing smart environments blending physical and virtual objects, introducing new opportunities in supporting individual and cooperative human activities. Augmented Reality can be defined as a medium in which digital information is added (superimposed) to the physical world in registration with the world itself and displayed to a user dependently on its location and its perspective [1,14]. Such a registration may occur in different ways, e.g. by means of fiducial markers placed in the environment, perceived and processed through the camera(s) mounted on smart-glasses or devices (or even on smartphones), or directly exploiting the spatial information obtained by sensors mounted on the AR visor. Mixed reality (MR) refers more generally to the merging of real and virtual worlds to produce new environments and visualisations where physical and digital objects co-exist and interact in real time [22].

Y. Demazeau et al. (Eds.): PAAMS 2018, LNAI 10978, pp. 127–139, 2018.
https://doi.org/10.1007/978-3-319-94580-4_10

AR and MR technologies can have an important impact from an application point view, allowing for reshaping the environments where people work and live, rethinking the way in which they interact and collaborate [4,23]. In that perspective, it is interesting to consider how these technologies can be put in synergy with pervasive computing/ubicomp technologies [21,24], and then Internet of Things (IoT) and Web of Things (WoT) as main ingredients of modern smart environments and spaces [3,12,16].

Agents can play a key role in AR/MR for modelling and implementing holograms that need to feature an autonomous behaviour, eventually interacting with other holograms and with humans, as well as with the physical world where they are immersed. More generally, one can envision future agent-based mixed reality systems where agents can dynamically create and control virtual objects and holograms, along with the control of physical things, so as to devise new kind of smart environments [17]. Besides individual holograms, multi-agent systems and agent organisations can be fruitfully exploited to model complex and possibly open, large-scale systems of holograms, featuring some degree of coordinated, cooperative, social behaviour.

In order to flexibly exploit existing agent/multi-agent models and technologies to this purpose, we argue the need of proper conceptual and technological frameworks. In this paper, we describe a novel framework called *Augmented Worlds* (AW) to develop agent-based pervasive mixed reality systems. The framework provides a set of features that – as a whole – aims at representing a significant contribution with respect to the state-of-the-art, in particular:

- *Shared/multi-user worlds* – the framework allows for multiple human users being immersed in the same AW, sharing and possibly interacting with the same augmented (virtual) entities.
- *Bidirectional augmentation* – the framework is based on a notion of "augmentation" which is wider than the pure AR/MR one, by integrating also the pervasive computing perspective where augmentation is about enriching the physical environment with computational capabilities.
- *Open systems* – the framework makes it possible to develop agent-based AR/MR systems where agents dynamically join (and quite from) an AW and where the structure of the AW – in terms of set of virtual entities/holograms – can be dynamically changed by agents at runtime.
- *Heterogeneous agent (programming) technology* – the design of the framework allows for different, heterogeneous agent technologies to be used for programming the agents working inside the augmented world. Examples are ASTRA [8], Jason [6], JaCaMo [5].
- *Cognitive agency* – in spite of the support for heterogeneous agent technologies, the framework has been conceived to be particularly effective for cognitive agents, in particular to investigate the engineering of AR/MR systems based on BDI (Belief-Desire-Intention) multi-agents systems.
- *Enabling AR technologies* – the design of the framework allows for fully reusing and exploiting existing AR enabling technologies – both software (e.g., Unity 3D, Vuforia) and hardware (e.g., Hololens, Meta2) – yet keeping a strong separation between such enabling level and the agent level.

– *Generality* – the framework aims at being sufficiently general to support the development of effective AW for different application domains—including both indoor and outdoor scenarios.

The remainder of the paper is organised as follows: Sect. 2 provides an overview of the conceptual model of AWs and of a concrete framework implementing the idea; Sect. 3 shows an example of AWs development and Sect. 4 describes the case studies adopted so far to start to discuss the idea. Finally Sect. 5 provides an overview of related works and some concluding remarks.

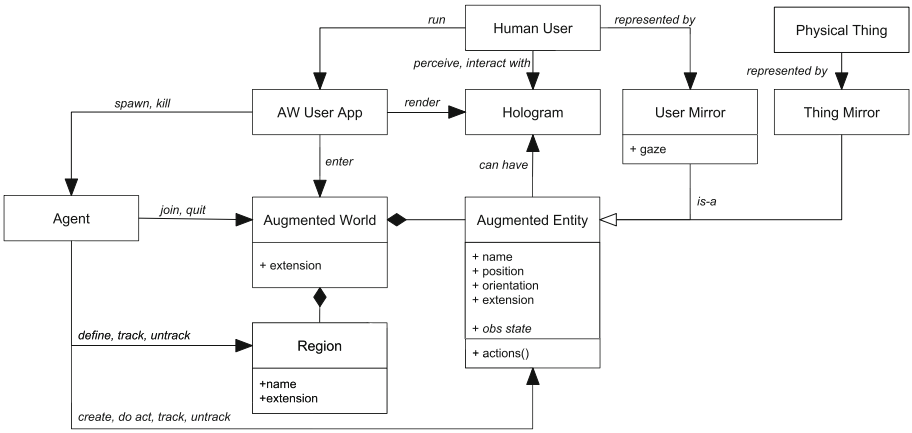

Fig. 1. Conceptual model of an augmented world.

2 The Augmented World Model and Framework

The conception of AW has been strongly influenced by the research in environments as first-class abstraction of MAS [25] and our experience with *mirror worlds* [17]—which are strongly related to AW. Accordingly, we model an agent-based pervasive mixed-reality system in terms of a dynamic set of software agents – possibly running on different hosts, including human wearable/mobile devices – working inside an application environment, possibly distributed as well. The application environment is composed by a dynamic set of virtual objects referred as *augmented entities*. Augmented entities are the basic bricks to shape the mixed reality environment, embedding some observable state and computational behaviour, and being situated in a specific position of the physical space. Agents can dynamically create, control, observe augmented entities. Some of these entities can have an AR representation, that we refer as *hologram*, that can be (shared and) perceived by human users immersed in that physical environment (by means of suitable AR devices).

The conceptual model has been strongly inspired by the A&A meta-model [15]. In fact, augmented entities can be conceptually modelled as *artifacts*

Table 1. Augmented worlds main primitives list.

Primitive actions	Description
joinAW(name, location): awID	To join an existing augmented world, getting an id of the session
quitAW(awID)	To quit from working in an augmented word
createAE(awID, name, template, args, config): aeID	To create a new augmented entity in a specified augmented world, specifying its name, template, parameters (that depend on the specific template), and initial configuration (including position, orientation, ...)
disposeAE(aeID)	To dispose an existing augmented entity
trackAE(aeID)	To start tracking an existing augmented entity
stopTrackingAE(aeID)	To stop tracking an existing augmented entity
moveAE(aeID, pos, orientation)	To change the position and orientation of an augmented entity, if allowed
defineRegion(awID, name, region)	To define a named region, specifying name and extension
trackRegion(awID, name)	To start tracking a region
stopTrackingRegion(awID, name)	To stop tracking a region

in A&A, used as first-class abstraction to design and develop the application environment where agents are situated. Beyond this, the AW conceptual model and framework introduce further concepts and features, detailed in the following, specifically tailored for pervasive mixed reality systems, such as – for instance – the spatial coupling with the physical reality and the explicit modelling of human users—which are not part of the A&A meta-model.

2.1 Conceptual Model

The main concepts of AW are formalised in the UML diagram shown in Fig. 1. An Augmented World is a virtual/computational world mapped on to some specific physical region (region attribute).

Augmented Entities. An Augmented World is composed by a set of Augmented Entity, representing the concrete (virtual) computational objects layered on top of the physical world. An Augmented Entity has a specific position, orientation and extension in the physical world (position, orientation, extension attributes), which are defined with respect to the region and system of reference defined by the Augmented World. Augmented entities are the basic bricks of the agent application environment. An Augmented Entity exposes an observable state (obs state attribute) – in terms of a set of observable properties, that can change

over time – and an action interface (actions() operations). Augmented entities are can be dynamically created and disposed by the agents, as well as moved in different point of the augmented world. **Agents**. In order to work inside an Augmented World, an agent has to *join* it, setting up a working session. Once joined an Augmented World, an agent can act upon an Augmented Entity, through the actions that the Augmented Entity makes it available, and *track* it, to perceive its observable state and events. Besides tracking specific augmented entities, agents can track *regions* inside Augmented World, as portions of the physical space on which the Augmented World is mapped. As soon as an augmented entity enters or leaves a region tracked by an agent, the agent perceives a corresponding event, carrying on information about the Augmented Entity source of the event.

The set of actions that an agent can do inside an AW can be grouped in two main categories. The first one is fixed set of primitives providing predefined functionalities (see Table 1): to join and quit an AW, to create and manage augmented entities, and to define and track regions. Besides this fixed set, by joining an AW the set of available actions is given by the collection of all action interfaces provided by the augmented entities actually created in that AW. So given an augmented entity aeID providing some action (operation) op, an agent has an action: doAct(aeID,op,args) which triggers the execution of the operation on the augmented entity. As in the case of environment programming abstractions [18], actions can be long-term processes whose execution occurs inside the augmented entity, eventually completing with a success or a failure. These action events are perceived by the agent triggering the action asynchronously.

On the perception side, in an AW an agent can track either specific augmented entities (`trackAE` primitive) or regions (`trackRegion`). By start tracking an Augmented Entity, an agent continuously receives percepts about its observable state and events generated. By tracking a region, an agent perceive events related to augmented entities entering or exiting, dynamically discovering their identifiers.

In cognitive agents based on the BDI model, beliefs about the state of the Augmented World and Augmented Entity are automatically created/updated/removed depending on what an agent is tracking.

Holograms. The bridge with AR is given by the Hologram concept (see Fig. 1). An augmented entity can have an *hologram* associated: the hologram is the AR representation of the Augmented Entity, to be perceived by the human users. This representation can be as simple as a 2D text, or a complex 3D structure. The key point is that – in an AR perspective – this representation is anchored (*registered*, in AR terms) to the physical world. The hologram model (state) depends on the state of the corresponding Augmented Entity and is kept updated every time the Augmented Entity state is changed. The hologram view – i.e., the actual rendering in the AR view – depends on the hologram state and the specific capabilities of the AR devices used by the user. Besides the representation, a hologram is what enables the interaction with the human users, in terms of gesture and gaze.

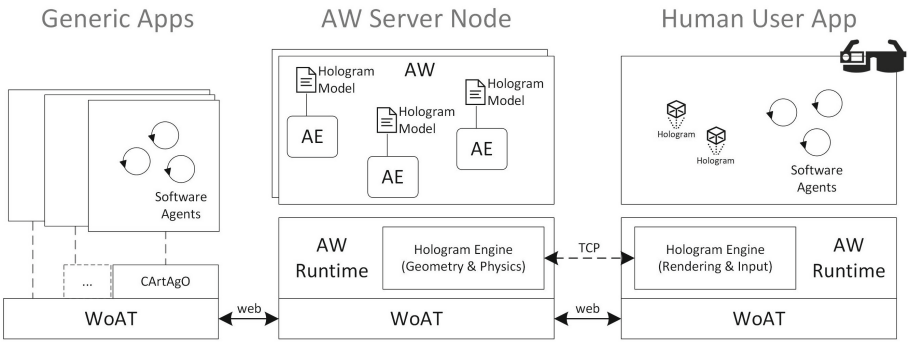

Fig. 2. Logical architecture of the AW framework.

Users. A human user starts a session inside an Augmented World by means of an AW user applications (AWUserApp). A AWUserApp spawn the agents that concretely can act inside the AW. From an UI point of view, the AWUserApp is responsible to create an AR view, given the current position and orientation of the user. Besides, this is the part in charge of detecting user inputs (e.g. gazing, gestures, voice commands). Such commands could be targeted towards specific holograms (e.g., grabbing a hologram, gazing a hologram). A hologram is then a source of observable events that can be perceived by agents, tracking the corresponding Augmented Entity. Multiple human users can be immersed in the same Augmented World, possibly using different AWUserApp.

Physical World Coupling. Some Augmented Entity can be used to represent physical objects (that are part of the physical environment) in the Augmented World, as a kind of mirror, so that the observable state of the Augmented Entity represents a model of the (physical) state of the physical thing. In Fig. 1 this is represented by the Mirror Thing concept, which is a specialisation of Augmented Entity. The coupling between the two levels – so that e.g. the observable state of the Augmented Entity is updated as soon as the physical one changes – is realised by proper *augmented entity drivers* (similar to device drivers), typically exploiting proper sensors and embedded technology. Then, by tracking an augmented entity coupled with some physical thing, an agent can perceive the physical state of the thing—as abstracted by the Augmented Entity. Among the physical things that can be coupled to an Augmented Entity, a main case is given by the physical body of a human user, so that agents can track the position of human users. This is represented by the User Mirror concept, which is specialisation Augmented Entity like in the case of Mirror Thing. The coupling between Augmented Entity and physical things can work also in the opposite direction, that is actions on the Augmented Entity can be useful to have an effect on physical things, by exploiting proper actuators. So in this case an agent can have an effect upon a physical thing of the environment by acting on the corresponding Augmented Entity.

2.2 A Prototype Framework and Infrastructure

On the basis of the AW conceptual model, we designed a first prototype framework and infrastructure for developing and running agent-based AR/MR systems. Figure 2 shows a representation of the logical architecture of the infrastructure.

From a structural point of view, we can logically recognise three main components:

- The **AW-Runtime**, which provides the infrastructure to execute AW instances, managing the augmented entities execution and providing a common interface for agents to interact and observe them. The runtime is designed to run on a server node (possibly distributed in cloud architecture), hosting the execution of one or multiple instances of AWs.
- the **Hologram Engine**, which supports holograms visualization and keeps updated their geometries in all human users devices, considering physics and managing inputs/actions of users over holograms. The main task of this component is to keep holograms (their geometries and properties) updated and consistent in each user application that wants to interact with the AW. It wraps and exploits enabling AR technologies. It runs both on the AW node and on each user device, directly communicating by means of e.g. TCP channels. From the AW point of view, the Holograms Engine provides the support to design the geometry of each hologram and keeps updated its representation according to entities properties. From the user perspective it provides all features needed to display the hologram according to the Augmented/Mixed Reality view, to manage physics (e.g. collisions between two holograms or perspective issues between holograms and real objects), and to manage user inputs and interactions with holograms—e.g. allowing to exploit gaze as a form of interaction or detecting hands/fingers gestures to manipulate holograms.
- the **WoAT** layer, which enables full interoperability at the application level. The architecture has been conceived by considering the requirements defined in Sect. 1, in particular: *(i)* full interoperability and openness at the application level, to be open to any technology for implementing agents working inside augmented worlds, yet enforcing an agent-oriented level of abstraction at the design level; *(ii)* distribution and scalability in the model adopted for defining and structuring the set of augmented entities. *(iii)* clean integration with the Internet of Things world. To this purpose, we adopted the Web of Things model proposed in the context of IoT [10], so that augmented worlds and entities (from an agent point of view) are network reachable resources providing a Web REST API to interact with them. That is, the interaction with augmented entities is mediated by a RESTful interface based on self-descriptive messages, HTTP operations (GET, POST, PUT, DELETE, HEAD) and event-oriented mechanisms like, e.g., web sockets. Each entity can be reached by means of a proper URL – related to the AW root URL – and (1) GET operations can be used to retrieve observable properties while

(2) POST operations can be used to send actions to be executed on the augmented entity. This interaction layer is referred as *Web of Augmented Things* (WoAT) [9]—the bottom layer in Fig. 2.

Current prototype is available[1] as open-source technology. The AW-Runtime is written in Java, using Vert.x[2] library to implement (part of) the WoAT layer. The Hologram Manager is based on Unity 3D equipped with the Vuforia plugin to manage AR aspects and with built-in scripts developed in C# to provides specific features for AW. A Java-based API is provided for developing the template of the augmented entity, allowing to create many instances of the same template. A taste of the API is provided in next section.

On the agent side instead, the AW any agent platform can be adopted to develop and run agents—this to promote interoperability and the vision of Web of (Augmented) Things. The Environment Interface Standard (EIS) [2] can be used to create the interface between existing agent programming platforms (e.g. GOAL, ASTRA) and the AW platform. In our case, we adopted the JaCaMo [5] platform to develop and run examples and case studies, described in next sections. Accordingly, the cognitive BDI-based agents are implemented in Jason and a set of artifacts – based on CArtAgO – has been exploited to ease the integration with the AW platform.

3 Developing AW with Cognitive Agents: A Taste

The aim of this section is to give a taste AW design and development, by considering a simplified version of a real-world case study, an *augmented museum* (discussed in next section). Figure 3 shows an abstract representation of the AW setting, along with the main architectural elements (agents and augmented entities) involved in the example, while Fig. 6 in next page shows a snapshot of the real-world setting. The AW is composed by an ancient Roman boat autonomously navigating over a virtual sea place among a set of real ancient amphorae, arranged in circle. If/when the boat hits a physical amphora, it goes back and changes direction. If/when the boat enters into a specific region ("red-zone") of the virtual sea, then a physical small light house switches on the light, which are switched off as soon as the boat lefts the zone. Multiple human visitors (users) can enter the AW, sharing the experience and interact with the boat. Visitors can generate some wind, affecting the trajectory of the boat. The coupling between the AW and physical world – i.e., the "registration" in AR terms – is based on Vuforia markers placed in environment, at the center of the stage.

Even if quite simple and classic, this example involves some of the main key points about AW: holograms featuring an autonomous behaviour interacting with the physical world, bi-directional augmentation, multi-user system.

[1] https://bitbucket.org/account/user/awuniboteam/projects/AW.
[2] http://vertx.io.

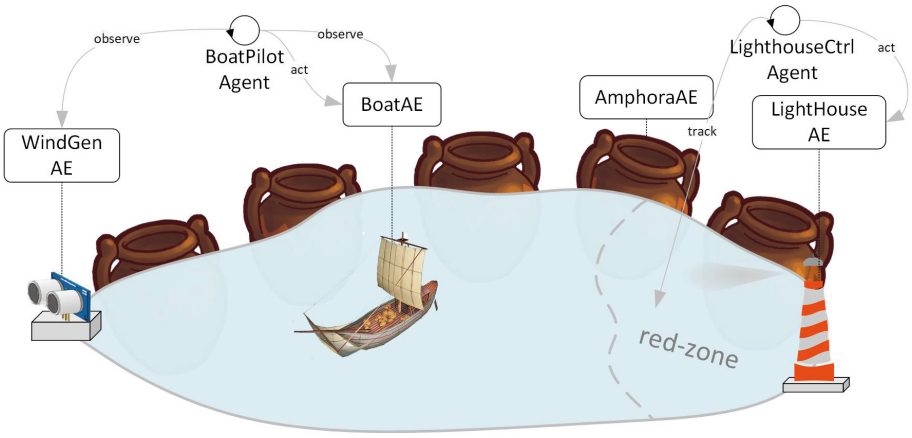

Fig. 3. Augmented museum example: AW design and main elements.

```
@HOLOGRAM("Boat")
class Boat extends AE {

  @PROPERTY double speedVal;
  @PROPERTY Vector2D speedVersor;
  @PROPERTY double windForce;

  @ACTION
  void setSpeed(Vector2D versor, double val) {
      customProperty("speedVal", val);
      customProperty("speedVersor", versor);
  }

  @ACTION
  void setWindForce(double windForce) {
    customProperty("windForce", windForce);
  }
}
```

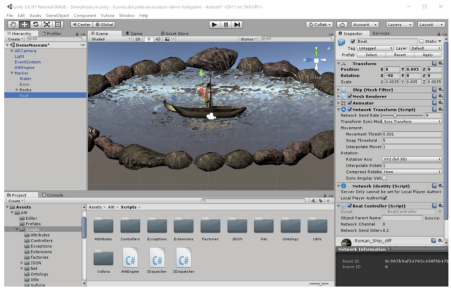

Fig. 4. (Left) The boat AE template definition. (Right) Unity screenshot with Boat prefab definition and details about AW C# scripts.

At the design level, a boat pilot agent BoatPilot encapsulates the autonomous behaviour of the boat, while the body and hologram of the boat is represented by the BoatAE augmented entity. Figure 4 shows a snippet of the source code of the BoatAE implementation based on current AW java-based API. The template of an augmented entity – defining the structure and behaviour of the instances – is mapped onto a Java class, properly annotated. In particular, `@HOLOGRAM`, `@PROPERTY` and `@ACTION` annotations can be used to identify, respectively, the Unity 3D prefab defining the geometry of the hologram, the observable properties and the actions of the entity. In the example, the BoatAE augmented entities have an hologram called Boat—shown in Fig. 4, right side.

Figure 5 shows a snippet of the code of the BoatPilot agent, implemented in Jason. The goal of the agent is to continuously move the boat according to a

```
!init.

+!init
  <- joinAW("museumAW"),
     trackAE("boat"),
     trackAE("wind"),
     !navigate(30).

+!navigate(Speed)
  <- getRandomOrientation(Versor),
     setSpeed(Versor, Speed),
     .wait(5000),
     !navigate(Speed).

@manage_border_collision[atomic]
+borderReached
  <- !divertBoatRoute(180).

+windForce(V)
  <- setWindForce(V).
```

```
!init.

+!init
  <- joinAW("museumAW"),
     trackRegion("museumAW","red-zone").

+regionUpdate("red-zone", "enter", AE)
  <- turnOnLighthouse,
     registerRedZoneAccess(AE).

+regionUpdate("red-zone", "exit", AE)
  <- turnOffLighthouse,
     registerRedZoneExit(AE).
```

Fig. 5. A snippet of the Jason source code of the BoatPilot agent *(left)* and of the *LighthouseController* agent *(right)*.

random trajectory and react to relevant events. The agent tracks (observes) BoatAE so as to react to a collision event. The collision event is generated when the BoatAE collides with any AmphoraAE augmented entity, which are augmented entities modeling/coupled to the real amphorae. The agent tracks also WindGenAE, to react to changes into wind force and acts on BoatAE accordingly. This WindGenAE augmented entity is coupled to a physical Thing, embedding an Arduino single-board micro-controller with a proximity sensor, to detect the distance of user hands, simulating the wind. In a Web-of-Thing style, this thing can be accessed using a REST web API—used in the WindGenAE implementation. To that purpose, the embedded system uses a Bluetooth connection to a gateway, based on Raspberry PI 3, functioning as a bridge to the Internet.

The MAS includes also the LightHouseControllerAgent (Fig. 5, right), which is tracking the region corresponding to the red-zone and switches on/off the light

Fig. 6. The snapshots from the Augmented Museum—a visitor watching the boat moving in the virtual sea placed within a set of ancient amphorae arranged in circle.

by acting on the LightHouseBuildingAE augmented entity. Also this augmented entity is coupled to a physical thing, based on Arduino. This agent reacts to the entrance/exit of any augmented entity (the boat in this case) and performs switchOn/switchOff action on the augmented entity.

4 Case Studies and Discussion

The augmented museum case study is part of a cultural heritage project, in cooperation with an Italian Museum in the context of maritime archaeology[3]. Generally speaking, the objective of the project is to investigate the design of augmented worlds layered upon two existing physical environments – a room in a museum and an harbour – so as to enrich the experience of the visitors. In the project involves two case studies: the augmented museum[4] – useful for us to start applying concretely the idea in the case of *indoor* scenarios – and the *augmented harbour*), which involves an *outdoor* scenario.

The augmented harbour – whose development is ongoing – accounts for designing an augmented world enriching a physical harbour with a group of virtual old fishing boats moving around in the sea and periodically entering the harbour. Visitors are meant to enjoy the exhibition from some fixed sightseeing points placed along the harbour side. Each sightseeing point includes a tablet with the camera pointing the (augmented) sea, allowing for some rotation to look around. In this scenario, the physical position of the boats as augmented entities with a hologram is given by a GPS device. As in the other case study, each boat is controlled by a pilot agent with the goal to mange boat route avoiding to bump other virtual boats entering and navigating in the harbour and avoiding to collide with harbour borders. To detect such collisions, the harbour borders are explicitly represented inside the AW by means of mirror augmented entities. This case study is useful to stress more the multi-agent aspect—possibly involving several boats moving around interacting among them and with the physical world.

Indeed, these are quite simple and "classic" examples of mixed/augmented reality systems. Currently we are looking for case studies to better stress aspects such as: dynamism—with agent dynamically manipulating (creating, disposing) virtual objects in the mixed environment; complexity of the cognitive agency—involving more complex autonomous behaviours for the agents; and user interaction—exploiting in particular the possibility for agents inside the AW to track user position and gaze.

5 Related Works and Conclusions

In literature, a comprehensive description about the role of agents and MAS in Mixed and Augmented Reality contexts has been already given with the

[3] Museo della Regina, Cattolica (RN), Italy.
[4] https://goo.gl/avBSgH.

concept of MiRa (Mixed Reality Agents) [11] and AuRAs (Augmented Reality Agents) [7]. The focus of the investigation of MiRA and AuRA is about agents having a representation in an AR/MR environment making them perceivable by human users and enabling interaction with them, as well as with other agents. Our work is aimed instead at defining a suitable conceptual model and technological framework for developing and executing agent-based AR/MR systems, featuring various degrees of distribution, autonomy, interaction with the physical world and flexibly exploiting (cognitive) agents as the key building block to design the autonomous parts.

Another main related work is given by Mirror Worlds [17,19], which can be considered the root of the AW idea. The AW conceptual model (described in Sect. 2) is an extension and evolution of the MW one, towards its concrete adoption in the design and engineering of real-world agent-based AR/MR systems. Besides, the design of the AW infrastructure is explicitly tailored to maximise interoperability with the Internet-of-Things as bridge to the physical world.

Finally, the literature on Intelligent Virtual Environments (IVEs) [13] is a further main related work of ours, in particular those contributions extending the basic IVEs towards the integration with the physical environment (e.g., [20]). Compared to these approaches, Augmented and Mixed Reality are treated as first-class aspects in the AW model and the platform, which is not the case of IVEs—more related to Virtual Reality. In this perspective, AW aims at fully enacting a form of *bi-direction augmentation* which is typically out of the scope of IVEs—integrating the Augmented Reality and Pervasive Computing, as supported by the Web of Augmented Thing idea and layer.

To sum up, the main aim of the AW framework is to provide a conceptual and practical tool for investigating and exploiting the use of cognitive agents and multi-agent systems for the development of complex augmented reality and mixed reality applications. In this paper we described *(a)* the AW conceptual model—which aims at being expressive enough to support the design of a variety of augmented worlds with different kinds of coupling between the virtual layer and the physical world; and *(b)* a first concrete platform, to explore in practice the development of AW for specific application domains. Future work will be devoted to refine and improve the architecture and prototype implementation of the framework, in particular by tackling more challenging case studies to get proper feedbacks to that purpose.

References

1. Azuma, R.T.: A survey of augmented reality. Presence: Teleoper. Virtual Environ. **6**(4), 355–385 (1997)
2. Behrens, T.M., Hindriks, K.V., Dix, J.: Towards an environment interface standard for agent platforms. Ann. Math. Artif. Intell. **61**(4), 261–295 (2011)
3. Billinghurst, M., Clark, A., Lee, G.: A survey of augmented reality. Found. Trends Hum.-Comput. Interact. **8**(2–3), 73–272 (2015)
4. Billinghurst, M., Kato, H.: Collaborative augmented reality. Commun. ACM **45**(7), 64–70 (2002)

5. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Sci. Comput. Program. **78**(6), 747–761 (2013)
6. Bordini, R.H., Hübner, J.F., Wooldrige, M.: Programming Multi-agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology. Wiley, Hoboken (2007)
7. Campbell, A.G., Stafford, J.W., Holz, T., O'hare, G.M.: Why, when and how to use augmented reality agents (AuRAs). Virtual Real. **18**(2), 139–159 (2014)
8. Collier, R.W., Russell, S., Lillis, D.: Reflecting on agent programming with AgentSpeak(L). In: Chen, Q., Torroni, P., Villata, S., Hsu, J., Omicini, A. (eds.) PRIMA 2015. LNCS (LNAI), vol. 9387, pp. 351–366. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25524-8_22
9. Croatti, A., Ricci, A.: Towards the web of augmented things. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 80–87, April 2017
10. Guinard, D.D., Trifa, V.: Building the Web of Things. Manning, New York (2016)
11. Holz, T., Campbell, A.G., O'Hare, G.M.P., Stafford, J.W., Martin, A., Dragone, M.: Mira-mixed reality agents. Int. J. Hum.-Comput. Stud. **69**(4), 251–268 (2011)
12. Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M.: Industry 4.0. business & information. Syst. Eng. **6**(4), 239–242 (2014)
13. Luck, M., Aylett, R.: Applying artificial intelligence to virtualreality: intelligent virtual environments. Appl. Artif. Intell. **14**(1), 3–32 (2000)
14. Milgram, P., Kishino, F.: A taxonomy of mixed reality visual displays. IEICE Trans. Inf. Syst. **E77-D**(12), 1321–1329 (1994)
15. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Auton. Agents Multi-agent Syst. **17**(3), 432–456 (2008)
16. Panetta, K.: Exploring augmented reality for business and consumers. Gartner report (2017)
17. Ricci, A., Piunti, M., Tummolini, L., Castelfranchi, C.: The mirror world: preparing for mixed-reality living. IEEE Pervasive Comput. **14**(2), 60–63 (2015)
18. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Auton. Agent. Multi-agent Syst. **23**(2), 158–192 (2011)
19. Ricci, A., Tummolini, L., Piunti, M., Boissier, O., Castelfranchi, C.: Mirror worlds as agent societies situated in mixed reality environments. In: Ghose, A., Oren, N., Telang, P., Thangarajah, J. (eds.) COIN 2014. LNCS (LNAI), vol. 9372, pp. 197–212. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25420-3_13
20. Rincon, J.A., Poza-Lujan, J.L., Julian, V., Posadas-Yague, J.L., Carrascosa, C.: Extending MAM5 meta-model and JaCalIVE framework to integrate smart devices from real environments. PLoS ONE **11**(2), 1–27 (2016)
21. Satyanarayanan, M.: Pervasive computing: vision and challenges. IEEE Pers. Commun. **8**, 10–17 (2001)
22. Schmalstieg, D., Höllerer, T.: Augmented Reality: Principles and Practice. Addison-Wesley, Boston (2015)
23. Starner, T.: Project glass: an extension of the self. IEEE Pervasive Comput. **12**(2), 14–16 (2013)
24. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. **3**, 3–11 (1999)
25. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Auton. Agent. Multi-agent Syst. **14**(1), 5–30 (2007)