Augmented Reality Web Applications with Mobile Agents in the Internet of Things

CONFERENCE PAPER · SEPTEMBER 2014

DOI: 10.1109/NGMAST.2014.24

6 AUTHORS, INCLUDING:

Teemu Leppänen
University of Oulu
21 PUBLICATIONS **51** CITATIONS
SEE PROFILE

Arto Heikkinen
University of Oulu
13 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Erkki Harjula
University of Oulu
39 PUBLICATIONS **229** CITATIONS
SEE PROFILE

Jukka Riekki
University of Oulu
190 PUBLICATIONS **1,193** CITATIONS
SEE PROFILE

Augmented Reality Web Applications with Mobile Agents in the Internet of Things

Teemu Leppänen*, Arto Heikkinen†, Antti Karhu†, Erkki Harjula†, Jukka Riekki* and Timo Koskela†
*Department of Computer Science and Engineering, University of Oulu, Finland
email: {teemu.leppanen, jukka.riekki}@ee.oulu.fi
†Center for Internet Excellence, University of Oulu, Finland
email: {arto.heikkinen, antti.karhu, erkki.harjula, timo.koskela}@cie.fi

***Abstract*—Augmented reality (AR) is a promising technology for building applications in an Internet of Things (IoT) environment, utilized for visualizing information provided by IoT devices. In this paper, we enable Web-based mobile AR applications with mobile agents in a resource-oriented IoT system architecture. We present an adaptable mobile agent composition that contains the data representation logic and mappings between AR applications and system resources. Thus, mobile agents and AR application-specific data structures are exposed as global system resources. System resource linkages are considered between real-world objects and their virtual representations for mobile agent-based AR applications. The agent composition also complies with the REST principles for resource access and control system-wide. This allows dynamic runtime adaptation and dealing with the device and resource heterogeneity, thus eliminating the need for application-specific communication protocols. Moreover, we utilize a Web-based mobile AR application framework, running completely in a Web browser, which facilitates straightforward AR application development. Lastly, a proof-of-concept mobile AR application is implemented, where a coffee maker with a visual tag is connected to a low-power resource-constrained wireless sensor network node as an IoT device. A mobile agent is injected into the IoT environment to expose state changes of the coffee maker. Through the visual tag, AR applications are able to visualize the state changes of the coffee maker in their user interface.**

***Keywords*—Augmented Reality, Web Services, Mobile Computing, Mobile Agent, Embedded Software**

I. Introduction

Augmented reality (AR) supplements physical reality by superimposing digital information upon the view of the real world [1]. In AR applications, the digital information is superimposed either based on the visual cues (e.g. an AR tag) found in the video stream of the real world or based on the location and/or orientation of the AR device itself [2]. Today, mobile devices are embedded with cameras as well as with location and orientation sensors which have made them a very potential platform for developing mobile AR (MAR) applications. However, the challenge with mobile devices is the heterogeneity of their hardware and operating system configurations. Furthermore, MAR applications typically utilize multiple distributed data sources from the fluctuating and evolving environment. Thus, the challenges in the MAR application development also include context-aware real-time information retrieval, object recognition and tracking, information visualization and user interactions [3].

Internet of Things (IoT) refers to globally connected interactive network of physical and virtual devices integrating disparate technologies and distributed intelligence [4]. The rapid development of IoT technologies has brought computing resources available in the surroundings of our everyday life. These computing resources include sensors that collect data from the environment and actuators that interact with the environment. MAR technologies provide an enriched user interface (UI) to monitor and control this kind of smart environments through augmenting digital information upon the view of the real world. The device and data heterogeneity is particularly important challenge for MAR applications in the IoT systems, where Web technologies has been suggested for human-machine interactions. To overcome the need of multiple standalone MAR applications, cross-platform supported Web browser can be used as a development platform [5].

Mobile code, i.e. software transferred between systems across a network, can offer several benefits for MAR applications: application-specific functionality, flexibility in designing system functionality, scalability as real-world objects can be accessed locally, interoperation capabilities with both virtual and real-world items, and furthermore, the mobile code implementations can be self-sufficient with data and execution state included [6]–[8]. Mobile code-based AR applications can become active participants in highly distributed systems, with access to both virtual and real-world objects [8], and capable of adapting in a dynamic system environment without user intervention and also without explicit reprogramming.

Mobile agents, as one of the mobile code paradigms, are autonomous programs that are able to control their own execution and migrate between devices in distributed systems. Mobile agents operate by executing tasks in the current host device, migrate their task, data and state into a new host device and continue their execution from the saved state in the new host device. Furthermore, mobile agents are able to access local system resources and infrastructure services.

This is the accepted version of the work. The final version will be published in the 8th International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST2014), September 10-12, Oxford, UK, 2014.

Therefore, with mobile agents, AR applications become active participants, capable of utilizing and controlling real-world objects through their virtual representations.

The contributions of this paper are the following: (1) an adaptable composition for the mobile agents is presented, capable of autonomous application-specific information processing and representations for AR applications; (2) resource-oriented IoT system architecture is presented, where mobile agents are active global system resources, breaking the coupling between AR applications, AR tags and mobile code; (3) use cases for mobile agents in AR applications are discussed; and (4) a Web-based proof-of-concept prototype MAR application with mobile agents in IoT is presented.

The rest of the paper is organized as follows. In the next section, we present the related work. In Section 3, we describe the overall system architecture. Section 4 presents AR-based interactions and use cases for mobile agents. Section 5 describes the proof-of-concept MAR application utilizing the presented system IoT architecture. Lastly, in Section 6, we discuss the benefits of our approach.

II. Related Work

Some experimental real-world application scenarios combining AR and IoT technologies have been presented. In [9], a prototype MAR system for visualizing the environmental sensor-based information is proposed. Another paper [10] uses a lightweight service architecture with AR application for analyzing aggregated sensor data from a factory environment. In [11], a mobile fitness game platform utilizes AR and sensors to interface virtual game content within the real world. These examples demonstrate how the AR applications in IoT assist firstly in collecting and processing large volumes of sensor data, and secondly, in presenting refined information to users in an understandable way.

In [6], the authors describe a mobile code-based approach for AR applications, based on battery-operated active tags containing links to the mobile code. When an AR tag is accessed, the code is downloaded using short-range communication protocol into an AR application, where the virtual representation of the real-world object is created based on the mobile code. However, a system specific communication protocol was developed for interactions to/from the mobile code. Vidgets framework [7] for MAR applications utilizes a controller application component to recognize visual tags and to map the tags with application descriptions, which contain a control interface description and a download address for the mobile code. When downloaded, the mobile code then handles the UI events and interacts with the system with a specific communication protocol. The system architecture includes separate a marker repository and a system service to host the mappings between the markers, applications and mobile code. UbiAgents [8] are animated mobile agents for AR applications, capable of changing their visual appearance and behavior. The composition of an AR agent is partitioned: (1) a centralized agent, in a knowledge-base, contains the control and reasoning logic, and is responsible for maintaining the agent state and its representations; and (2) a mobile agent for the representation migrates into the application platforms. Agent tasks are described as goals, decomposed into low-level commands for execution and transferred as a part of the agent representations to the applications. Device schemata in the knowledge base is utilized for exposing device capabilities and interface descriptions to the system. Notifications from the devices are stored into the knowledge base, as a centralized communication medium. When a device reports a state change, observers such as AR applications update their UI and agent controllers control of the agent migration automatically.

We extend the work in mobile agent-based AR applications by considering the mobile agents, the AR applications and the IoT devices as shared resources in a loosely-coupled distributed IoT system, with their dynamically available capabilities and application-based functionality. We introduce adaptable mobile agent composition for AR applications, which contains the mappings between applications and required system resources. In this work, the HTTP methods are simple communication primitives and eliminate the need for specific communication protocols. The development frameworks that enable implementation of MAR applications on a web browser are very few [5], [12]. Apart from our web-based development framework for MAR applications, the existing solutions do not consider mobile agent-based communication and cooperation aspects in the IoT.

III. System description

The main components of MAR application are the computational platform for coordinating the tracking and coupling of physical and virtual worlds, displays to incorporate the virtual data in the physical world, portable input methods to interact in the augmented world, wireless networking to communicate with the infrastructure as well as the storage and access for the data [3]. The sensors, both infrastructure based and mobile, attached to the user's device with the various location-based and context-aware information services, are the essential building blocks and shared resources in the IoT systems.

In the Web, resource-oriented architectures (ROA) consider the resource as the main abstraction, exposed with a unique URI, with various representations of the state and RESTful interfaces for access. Here, HTTP offers standardized communication protocol with unified communication primitives over the Web. This provides high level abstractions for the resources and resource access, promotes simplicity in the application design, enables re-usability across applications and exploiting locality in AR applications, while maintaining scalability. In the AR application development, the same communication primitives facilitate information retrieval from both real-world and virtual objects, including the mobile agents, and controlling the real-world objects through the virtual representations.

The IoT devices are considered as system resources including their physical components (such as sensors), and their data and computational capabilities (such as processing power and memory). For AR applications, the Point-of-Interests (POI), associated with AR markers and metadata, are also exposed as system resources. This opens up new scenarios for utilization of POIs in IoT and AR applications. Furthermore, the mobile agents, with their application-based computations and the various state representations, are considered system resources. This enables addition of new functionality in the system in runtime and using resources as the building blocks

TABLE I: The mobile agent composition

Name	*Resource, i.e. the agent, name*	
Code	*Identifier*	*Program or reference (URL)*
	Programming language	Code block
	Programming language	URL
Resource	*Name*	*Reference (URL)*
	Variable ...	URL ...
State	*Name*	*Representation*
	Variable ...	URL ...
Metadata	*Element*	*Data*
	Access rights	...

for high-level services [13]. With mobile agents, the system functionality including the functionality of the AR application, is distributed among the IoT devices.

A. Mobile Agents

The mobile agent composition utilized in this work is based on our previous work in [13], which enables migration as a self-sufficient computational unit. The composition consists of several segments: name, code, resource and state. The name segment refers to the name part of the URL, thus the agent state can be accessed with the hosting device address and the name. The code segment contains the computation the agent executes in the selected programming language. If a reference is utilized, the code is not included in the composition, but downloaded before execution. The resource segment lists the system resources that the agent utilizes as URLs. A selection of resources is represented as local, referring to the IoT devices where this agent migrates, and remote, referring to the resources in external systems where the agent does not migrate but downloads the resource representations. The state segments presents the current, i.e. intermediate, result of the task as the resource representation. See Table I for illustration of the mobile agent composition.

In AR applications, the mobile agent composition can be utilized in the following way. As the code segment is platform and programming language independent and enables multiple code blocks, it can include the separated data processing and UI codes for different platforms and application-specific purposes. The identifier addresses the intended platforms in the IoT system and if a reference is used, the code can be downloaded separately from a repository. This could be useful with large-size agent compositions, and moreover, when the UI components or data processing functions are either generalized within the system or coupled with a specific AR tag. The resources are: (1) location-based, additionally coupled to the specific AR tag; or (2) remote information exposed in the IoT system and utilized in the agent computations and visualized in the AR application. The results of each data processing task are to be presented in the state segment and accessed separately by their resource names, which enables utilization of the agent as a service. If the UI code is incorporated with a specific state, the UI situation can be saved, migrated with the agent and continued later in any capable platform.

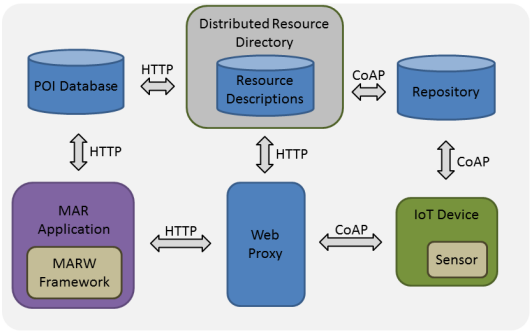

Fig. 1: IoT system architecture with MAR application

B. IoT System Architecture

The IoT system architecture is based on the IETF CoRE framework, where the components include heterogeneous low-power resource-constrained embedded devices with integrated physical sensors and displays. A proxy component enables application-specific functionality, a resource directory operates as a name server and repositories are used for information storage.

Figure 1 illustrates our IoT system architecture. The Web proxy abstracts the heterogeneous IoT resources over disparate networks for unified Web access and operates as a protocol translator between communication protocols such as HTTP and CoAP [13]. The Web proxy can expose application-specific functionality as services with proprietary interfaces.

The distributed resource directory (DRD) [14] is a name server storing the global system resource descriptions, which include the resource URI and name, the address of the current hosting device, the content-type of the data and the related metadata such as a semantic description and the access rights. The resources can be queried based on any of these elements. To join the system, each IoT device, application, system component or mobile agent needs to register its resources to DRD. DRD provides both HTTP and CoAP interfaces for communication over disparate networks. This implements the knowledge-base [8], required for runtime adaptation in AR applications in the fluctuating IoT system. DRD is presented as a logical entity containing all the system resources, but may be deployed as separate databases and repositories.

Repositories store the resources of MAR applications, such as POI entities with the associated AR tag information, system services, available IoT data sources, and dynamic functionality through the mobile agent compositions. System components can add computations, with access rights, in runtime to the repositories with a corresponding identifier for the programming language, intended operating system or device platform.

C. Framework for Augmented Reality Applications

To enable the development of MAR applications that are compatible with all mobile platform and operating system configurations, we implemented a web-based mobile AR development framework (MARW). MARW relies on the Web technologies including HTML5, WebRTC and XML3D. The

http://datatracker.ietf.org/wg/core/charter/

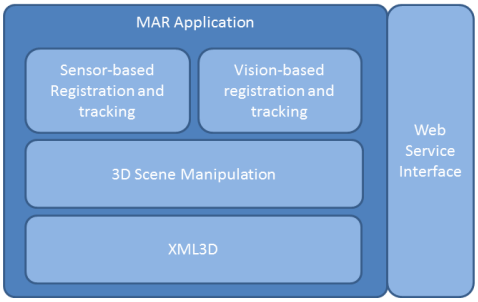

Fig. 2: Mobile Augmented Reality Framework

XML3D provides declarative representations of 3D content for HTML [5]. The modular implementation of MARW allows the AR application to utilize only the required components. See Figure 2 for illustration. As the implementation details of MARW will be presented in another paper, the functionality of each component is summarized only in brief here.

The sensor-based registration and tracking component is used for determining the location and orientation of the AR device based on the data obtained from multiple embedded sensors. Vision-based registration and tracking component is used for detecting visual features in a video stream obtained from the camera of the AR device. 3D scene manipulation component provides methods for converting the data acquired from sensor-based and vision-based registration and tracking components into a format that is compatible with the XML3D component, which manages the 3D elements that are augmented on the top of the video stream acquired from the camera of the AR device. The Web service interface provides a communication interface between the AR application and one or multiple third party Web services. It supports both a RESTful API over HTTP and a WebSocket connection.

D. Point-of-Interest Database

The POI database [15] stores information about geographic entities, and is here considered a part of the DRD, to enable utilization of POIs as system resources. A POI is presented as data model containing information on a single geographic entity that can be, for instance, a building or a restaurant, but also an IoT device. The POI data model is modular, consisting of different data components that serve different application needs. The POI data includes, for example, name, physical location, AR tag with a unique identifier and URLs for accessing the system resources related to a POI. See Table II for an example. System resource lookups to the POI database can be based on different criteria, such as name, physical location and AR tag identifier. Thus, the POI database is a repository, where the system resources are the POI descriptions and queried based on the AR tag identifier.

IV. AR-based Interactions and Use Cases for Mobile Agents

We consider two types of interactions. Figure 3 illustrates the interactions between a MAR application and IoT system components over disparate networks. The basic operation is conducted as follows: (1) tracking for AR tags; (2) selecting a POI entity based on the AR tag identifier; and (3) retrieving the resource representations listed in the POI data for visualization in the MAR application. From the user experience standpoint, the following set of design principles has been suggested for MAR applications [3]: (1) use the sensory data-based context to provide content; (2) enable users to focus on the relevant information; (3) demonstrate clearly the content privacy level in the application; (4) provide feedback about the application interactions with the various infrastructure services; and (5) enable smooth user interactions based on user's procedural and semantic memory.

In our IoT system architecture, when a request is received by the Web proxy for a particular system resource, the resource state is retrieved from the host. If the requested system resource is a mobile agent, the Web proxy will perform a lookup to DRD to see whether the mobile agent is already injected into the system. If the mobile agent exists, its state is retrieved from the hosting device. If the mobile agent has not been injected into the system, the Web proxy will interpret the request, look up the resources from DRD, create the mobile agent composition in runtime from the available resources, and inject it into the system [13]. Additionally, the mobile agent composition can be specifically injected into the system through the Web proxy, where the composition is included in the request from the AR application. For mobile agent-based linking of real-world objects with virtual representations, linkages were described in [16]: (1) a moving tag/reader is carried into a place, where detection of the existing tag/reader initiates agent migration from the current static host to the static host in place; and (2) a moving host with a tag/reader enters a place, where the agent migration from the moving host into the static host is initiated by detection of the tag/reader. By assuming the reader being the AR application, the tag being an AR tag and the hosts being IoT devices, we are able to implement these linkings with the presented IoT system architecture and adaptable agent compositions.

However, the previous work [6]–[8] considers the mobile agents as a UI component bridging AR applications and AR tags. Extending the work, we conceptually break this coupling as the mobile agent composition is an IoT system resource, facilitating autonomous and adaptive operation while providing application-specific state representations and resource control interfaces. The presented IoT system architecture facilitates loose-couplings between system resources [13]. Now, the mobile agent is neither tightly coupled into an AR tag nor an AR application, but utilizes both as a resource in its computations; the AR application platform for data processing and the AR tag as a resource identifier. Mobile agents are capable of virtually following the user in the physical world, while considering the contextually best IoT device for task execution. With AR applications, this is enabled by utilizing the detected POI as a system resource with a known location. In turn, the AR application can consider the mobile agent as any data source in its visualization task, and if required, control the mobile agent through the exposed interfaces. As the agent composition is adaptable, the system components can modify it in runtime; add/remove data processing functionality and even UI code, add/remove resources such as new data sources; and modify what is exposed by the mobile agent in its state. Moreover, autonomous mobile agents can live without the AR application in an IoT system as service providers.

This extends the possibilities for the mobile agent utilization in AR applications in IoT systems. Both weak and strong code mobility now become feasible as all the code, data and state can be migrated. As described, this requires dynamically updated DRD to cope with a fluctuating system environment and resource availability. Standardized and unified communication interfaces, accomplished with HTTP methods, enable the resource-constrained IoT devices to join in the IoT system and interact through the embedded Web services.

V. Proof-of-concept Application

We implemented a real-world prototype MAR application with our MARW framework based on the presented IoT system architecture. The MAR application was run in a Web browser in a Android tablet that was connected to the Internet over Wi-Fi. We utilized a WSN node with a power consumption monitoring sensor [17], communicating atop 6LoWPAN in 868Mhz frequency band, as an IoT device. The WSN node was connected to the Internet via a WSN router. In addition, a coffee maker was connected to the power plug in the WSN node, as shown in Figure 4.

The MAR application operates with the components provided by the MARW framework, namely the sensor-based and vision-based registration and tracking, the 3D scene manipulation, the XML3D and web service interface. An AR marker was attached to the coffee maker to enable its visual detection by the MAR application. To enable the visualization of the marker information, the marker was first registered into the MAR application by its identifier. As shown in 3, the MAR application can discover the system resources visible to the user by first recognizing the AR markers, and then querying the POI database for the corresponding POI entity with the AR marker identifier and the current location. See Table II for the POI entity description: information about the physical device such as the name, semantic description, physical location and the mobile agent state, the resource ”ar_demo”, as the data source. The mobile agent composition is shown in Table III, where the agent computation code is in Python and in the IntelHEX binary format, targeted for the WSN node [13]. The Web proxy performs a data format conversion between JSON and CSV used in the HTTP and CoAP communication, respectively.

The WSN node has registered its resource, i.e. data from a power monitoring sensor, exposed as a resource named ”power”, into DRD. This resource is utilized by the mobile agent to compute the coffee maker state as shown in 3. After creation of the mobile agent composition and injection into the system, it migrates into the WSN node for execution for its lifetime. Then, the WSN node registers the mobile agent as its resource into DRD, as in Figure 3. Now the system components can find the current host of the mobile agent and access the state. The state contains the thresholded power measure as ”on” or ”off”, the used threshold value and the timestamp of the last state change. From the state, the MAR application can visualize the ”freshness” of the coffee in terms of how long the coffee maker has stayed in the ”on” state. With one resource in the resource segment, the mobile agent does not migrate further. For the agent computation, we determined the power consumption profile from the sensor data in advance. Now, as the agent state has been established as a service in the system, any MAR application can now visualize the coffee maker state through the AR marker.

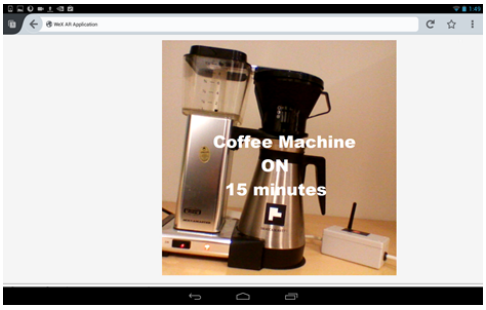

Fig. 4: MAR application UI in a Web browser

VI. Discussion and Conclusions

We extend the uses of mobile agents in AR applications by considering them as global system resources in IoT instead of being tightly coupled with AR tags or AR applications. The adaptable mobile agent composition enables both application-specific data processing and visualization components as a self-sufficient mobile code, capable of migrating in the IoT system based on the available resources. With the composition, the mobile agents can react to the fluctuating IoT environment in runtime and the system components are able to modify the composition based on their requirements [13].

As the presented IoT system architecture is based on the principles of ROA, we facilitate the development of Web-based AR applications in IoT. The MARW framework was used for implementing MAR applications in mobile Web browsers, where the communication is based on RESTful interfaces over HTTP. The MARW framework, with mobile agents, enables MAR applications in accordance with the given design principles [3]. (1) The device location is utilized in MAR applications through the device API, (2) the application-specific mobile agents facilitate exposing context-aware information, (3) the mobile agents expose the sensitive information according to the application-specific requirements, and (4) component interactions can be exposed also through the agent state. Contemplating smooth user interactions (5), mobile agents can offer personalized learning capabilities based on the derived user context and UI usage details.

A functional MAR application was implemented with 90 lines of JavaScript code with our MARW framework, including the POI and mobile agent handling. The mobile

TABLE II: The demonstration POI data structure

POI	id	...
Metadata	semantic name device category description location last update	Coffee Maker Appliance IoT Coffee Maker in Cafeteria [latitude, longitude] 27-05-2014T10:49:00
Resource	URL	http://www.proxy.net/ar_demo
Marker	id type	... 3x3px

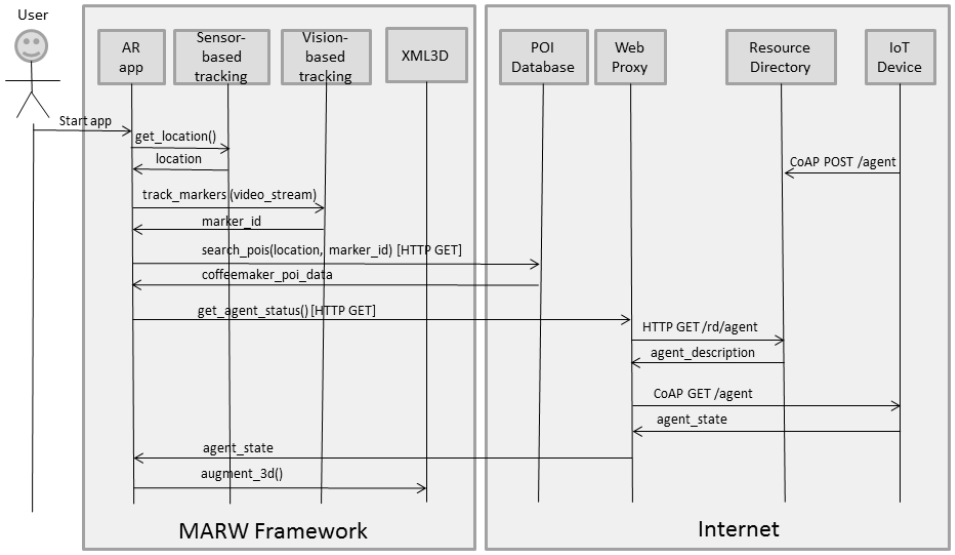

Fig. 3: MAR application and system component interactions

TABLE III: The demonstration mobile agent composition

Name	ar_demo	

Code	python	if coffee_ready == (power >threshold):
		time += 1
		else:
		coffee_ready = not coffee_ready
		time = 0

	IntelHEX	[...]
Resource	power	coap://sensor.wsn.net/pwr
State	threshold time ar_demo	500 0 [coffee_ready, threshold, time]

agent composition includes mappings to the required system resources and the computation results are exposed in the state segment. The mobile agent programming has been explained in our previous work [13], where the computation code can be implemented with platform-specific programming languages, however possibly requiring pre-compilation. The UI components are straightforward to add into the mobile agent composition by utilizing the mapped resources. Although the service content data formats can be negotiated with HTTP and CoAP capabilities, an additional step may be needed to convert between data formats before the resource representations can be utilized in the computations. This data format conversion is built-in to the common scripting languages, such as Python and JavaScript, for the most common content types in the Web.

The presented building blocks for MAR Web applications facilitate the use of IoT resources in widespread application scenarios for augmenting everyday life. As the future work, we aim to further extend the real world application scenarios with user experience studies.

ACKNOWLEDGMENT

This work was conducted in the MAMMotH project, funded by the Finnish Funding Agency for Technology and Innovation, and the FI-WARE project, funded by the European Commission. We thank Dr. Meirong Liu for the work with DRD and Mr. Pauli Närhi for the work with the WSN node.

REFERENCES

[1] R. Azuma *et al.*, "A survey of augmented reality," *Presence*, vol. 6, no. 4, pp. 355–385, March 1997.

[2] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy, and B. Davidson, "The argon ar web browser and standards-based ar application environment," in *10th IEEE Int. Symp. on Mixed and Augmented Reality*, 2011, pp. 65–74.

[3] P. E. Kourouthanassis, C. Boletsis, and G. Lekakos, "Demystifying the design of mobile augmented reality applications," *Multimedia Tools and Applications*, pp. 1–22, October 2013.

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, October 2010.

[5] F. Klein, D. Rubinstein, K. Sons, F. Einabadi, S. Herhut, and P. Slusallek, "Declarative ar and image processing on the web with xflow," in *Proc. of the 18th Int. Conf. on 3D Web Technology*. ACM, 2013, pp. 157–165.

[6] K. Kangas and J. Röning, "Using mobile code to create ubiquitous augmented reality," *Wireless Networks*, vol. 8, no. 2/3, pp. 199–211, March-May 2002.

[7] H. Kimura, E. Tokunaga, and T. Nakajima, "System support for mobile augmented reality services," in *Proc. of the ACM Symp. on Applied computing*, 2007, pp. 1616–1623.

[8] I. Barakonyi and D. Schmalstieg, "Augmented reality agents for user interface adaptation," *Computer Animation and Virtual Worlds*, vol. 19, no. 1, pp. 23–35, February 2008.

[9] D. Goldsmith, F. Liarokapis, G. Malone, and J. Kemp, "Augmented reality environmental monitoring using wireless sensor networks," in *12th Int. Conf. on Information Visualisation*, 2008, pp. 539–544.

[10] M. Berning, T. Riedel, Y. Ding, T. Miyaki, N. Fantana, and M. Beigl, "Augmented service in the factory of the future," in *9th Int. Conf. on Networked Sensing Systems*, 2012, pp. 1–2.

[11] J. Westlin and T. H. Laine, "Calory battle ar: An extensible mobile augmented reality exergame platform," in *IEEE World Forum on Internet of Things*, 2014, pp. 171–172.

[12] C. Oberhofer, J. Grubert, and G. Reitmayr, "Natural feature tracking in javascript," in *IEEE Virtual Reality*, 2012, pp. 113–114.

[13] T. Leppänen, M. Liu, E. Harjula, A. Ramalingam, J. Ylioja, P. Närhi, J. Riekki, and T. Ojala, "Mobile agents for integration of internet of things and wireless sensor networks," in *IEEE Int. Conf. on Systems, Man, and Cybernetics*, 2013, pp. 14–21.

[14] M. Liu, T. Leppänen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, and T. Ojala, "Distributed resource directory architecture in machine-to-machine communications," in *9th IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communications*, 2013, pp. 319–324.

[15] A. Heikkinen, A. Okkonen, A. Karhu, and T. Koskela, "A distributed poi data model based on the entity-component approach," in *19th IEEE Symp. on Computers and Communications*, 2014, [Accepted].

[16] I. Satoh, "Linking physical worlds to logical worlds with mobile agents," in *Proc. of the IEEE Int. Conf. on Mobile Data Management*, 2004, pp. 332–343.

[17] T. Leppänen, J. Ylioja, P. Närhi, T. Räty, T. Ojala, and J. Riekki, "Holistic energy consumption monitoring in buildings with ip-based wireless sensor networks," in *Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, 2012, pp. 195–196.