International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 3 (2011) pp. 771-779

Service-oriented and Agent-based Architecture Supporting Adaptable, Scenario-based and Context-aware Provision of Mobile e-Learning Services

Stanimir Stoyanov[1], Ivan Ganchev[2], Damyan Mitev[1], Vladimir Valkanov[1], Máirtín O'Droma[2]

[1] Department of Computer Systems, University of Plovdiv,
236 Bulgaria blvd., Plovdiv 4002, Bulgaria
stani@uni-plovdiv.bg; d.mitev@isy-dc.com; assarel@abv.bg

[2] Telecommunications Research Centre, University of Limerick,
Plassey, National Technological Park, Castletroy, Limerick, Ireland
{Ivan.Ganchev;Mairtin.ODroma}@ul.ie

***Abstract*: This paper describes an OMG's MDA-based approach for the development of a service-oriented and agent-based middleware architecture supporting flexible and adaptable, scenario-based and context-aware provision of mobile e-Learning services within InfoStation wireless environments. Considering the system development as a process of iterations, the approach provides an extensive ability to examine different development aspects and extend the system architecture step by step. The first two iterations, namely the base middleware architecture and the scenario-based management, are described in detail. A simulation environment used for testing the architecture is also presented.**

***Keywords*:** m-Learning, adaptable software architecture, MDA-based approach, intelligent agents, mobile access, InfoStation environment**.**

I. Introduction

A distinguishable feature of contemporary mobile e-Learning (m-Learning) systems is the anywhere-anytime-anyhow aspect [1] of delivery of electronic content, which is personalized and customized to suit a particular mobile user [2], [3]. In the light of this trend, our goal is to develop a software architecture which is able to support m-Learning services delivered within a University campus based InfoStation communication environment with distributed control. The InfoStation paradigm is an extension of the wireless Internet, where mobile clients interact directly with Web service providers (i.e. InfoStations). The users request services (by using their mobile devices) from the nearest InfoStation via available Bluetooth (IEEE 802.15), WiFi (IEEE 802.11), or WiMAX (IEEE 802.16) connections. In our approach, each application utilizing the InfoStation infrastructure consists of two components: (i) a standardized middleware, which identifies and locates the changes in the environment in order to prepare adequate delivery of requested services; and (ii) a set of electronic services (eServices) – mainly e-Learning services –, which are adapted and controlled by this middleware.

In the original InfoStation architecture [4], the InfoStations operate as mediators between the user mobile devices and an InfoStation Center, on which a variety of eServices are deployed and executed. However in our architecture, the InfoStations are not only mediators but primarily service-providing nodes. The implementation of such architecture raises serious challenges. The main one among these is related to the support of distributed control, which to be able to detect changes in the environment (*context-awareness*) and according to these changes to offer the requested services in more flexible and efficient way (*adaptability*).

In the past few years different context-aware systems for different purposes were presented in the literature. An architecture similar to our own architecture is presented in [5]. This context-aware architecture is composed of clients (moving nodes), context-server and middleware (fixed nodes connected through TCP/IP to the context-server). The middleware plays important role both in identifying the clients using the Bluetooth technology and in finding a suitable executable module in accordance with the context acquired from the context-server. Another architecture described in [6] uses a Context Engine for context-aware delivery of Web services by utilizing a rule-based approach based on first-order logic for centralized processing of context. In [7], a context broker architecture based on ontology for context representation is presented. A context-aware service provider – an analog to a telephone provider – is described in [8]. More common context-aware

architectures are presented in [9], [10]. The context in these architectures is stored and processed in a centralized way whereby the middleware is used mainly as a mediator.
In our architecture the context is processed mainly in the middleware components deployed on different InfoStations. The determination of a particular context is accomplished within the framework of predefined scenarios. To achieve this, in certain cases it is necessary to identify different local situations happening at different moments and in different network nodes (InfoStations). In cases like these, synchronization between the middleware components deployed on different InfoStations is required.

Another specifics of our architecture is that it is being developed as an agent-based one in order to:

- Model adequately a real distributed infrastructure;
- Allow for realization of distributed models for control;
- Show the pro-active behavior of the middleware (which is quite beneficial in many situations);
- Use more efficiently the information resources spread over different InfoStations.

Moreover, the agent-based architecture is implemented as a set of autonomous agents, which could be easily extended with new agents that communicate by means of a standardized protocol, i.e. the FIPA's Agent Communication Language (ACL), [11]. This differs from the classic multi-tier architectures in which the relations between components at the same tier are much stronger. In our architecture we use tiers too; however, these are mostly logical groupings of autonomous components based on their functional characteristics.

Another important problem is the context-aware provision of m-Learning services. To achieve context-awareness, the middleware must be able to identify/detect all changes occurring in the environment and adapt its behavior accordingly. Essential here is the concept of *context*. We use the Dey's definition [12], according to which "context is any information that can be used to characterize the situation at an entity". The entity could be a person, a place, or an object, which is considered relevant to the interaction between a user and an application, including the user and the application themselves. Context could be of different type, e.g. location, identity, activity, time. By utilizing the capabilities of the InfoStation infrastructure, in our architecture we want to ensure trouble-free, transparent and adequate execution of user requests for services by taking into account the changes occurring in regard to the user/device location, user mobile device, access network type (Bluetooth, WiFi, WiMAX), user identity, etc.

This paper presents our approach for the development of a middleware architecture that is consistent with the fundamental principles of the e-Learning systems development suggested by the IMS Global Education Consortium, such as interoperability, service-orientation, component usage, layering, behaviors and data modeling, multiple binding, and adoption [13]. The middleware is being developed step-by-step, whereby the steps are considered as iterations. The first version of the middleware architecture, which was developed by following this approach, is presented here. We also discuss an extension of the architecture aimed at better control of service requests' execution in different scenarios.

II. Approach

The development of context-aware, flexible and adaptive architectures is exceptionally difficult task, which is impossible to complete without a clearly-defined approach and without taking into account the fundamental requirements of the future system. For the development of our m-Learning system we used a software development approach [14] for facilitating the use of a specialized architecture (InfoStation-based). The approach aims at the development of a service-oriented and agent-based architecture for efficient use in on-line e-Learning systems employing an InfoStation infrastructure for getting mobile access to eServices and educational resources within a University campus [15], [16]. The approach is based on the ideas suggested by the MDA specification of OMG [17]. In conformity with this approach, an architecture was built on three levels (Figure 1):

- *eServices level* – represents and models (using a suitable formalism) the functionality of eServices provided by the system;
- *Middleware level* – an agent-based multi-layered level playing a mediation role between the eServices level and the scenarios level. It offers shared functionality: on one hand, it contains agents needed for the execution of different scenarios, and on the other hand, it specifies a set of agents assuring the proper provision of eServices;
- *Scenarios level* – presents the features of the InfoStation architecture and the context types supported by it (in the form of different scenarios), which are used for accessing the eServices.

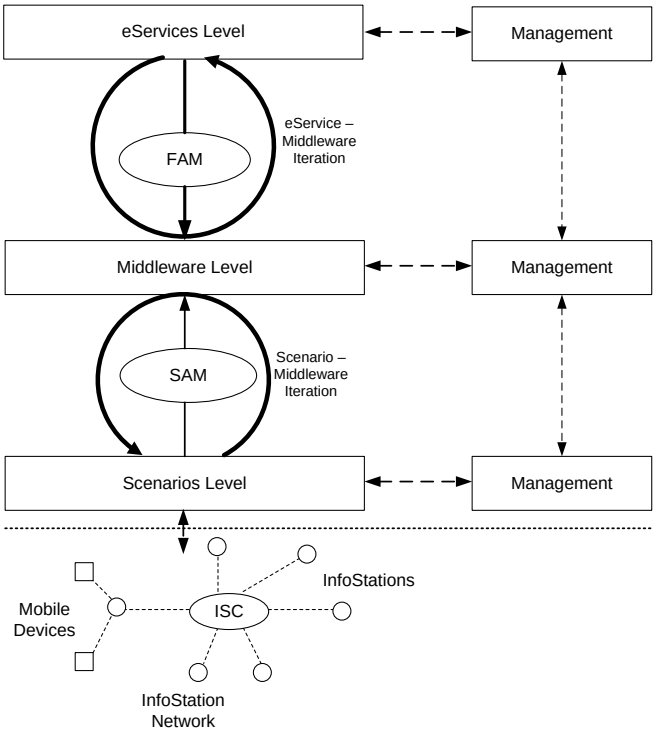

Figure 1. The levels and iterations of the approach used

In addition, we examine the system development as a process of iterations. The term *iteration* – borrowed from the Unified Software Development Process [18] – means a

workflow or cooperation between the developers at different levels so as to be able to use and share particular products and artefacts. We define two types of iterations in our approach: (i) between the scenarios level and the middleware level, and (ii) between the middleware level and the eServices level.

In accordance with this approach we suggest a hybrid middleware architecture built upon intelligent agents and eServices. The eServices are convenient for the implementation of specific business functionality, but are static, however. The necessary flexibility, adaptability and collaboration features of the architecture, required for the context-aware middleware, are ensured by intelligent agents (which, however, are unsuitable for the implementation of functionality). The agents spot any change in the communications environment, carry out the adaptation necessary for correct processing of service requests, and activate the desired eServices.

The following lower-level iterations are envisaged for realization in our architecture:

- '*Base Architecture'* – aimed to support the main scenarios characterizing the operational specifics of the InfoStation network [19], a base middleware architecture (presented in the next section) has been developed. More details on this can be found in [20], [21].
- '*Scenario-based Management'* – some important changes occurring in the user context during the execution of the users' service requests (e.g. the device mobility, when a user/device leaves the service area/range of one InfoStation and enters another) can be detected and identified by the system, only if the temporal aspect of this process is taken into consideration. Thus the goal of this iteration is to develop concepts and formal models, and to realize efficient scenario-based management of the offered eServices and active user sessions.
- '*Resource Deficit'* – in some cases the user requests for particular services cannot be satisfied fully by the local InfoStation due to a resource deficit (e.g. when the information needed to satisfy the service request is unavailable on the local InfoStation). In these cases the service provision must be globalized in a manner involving other InfoStations. The software needed to support this type of InfoStation's interaction is developed as part of this iteration.
- '*Adaptation'* – this iteration is concerned with problems related to strengthening the architecture, e.g. to support adaptability. In our opinion, personalized e-Learning process can be fully realized only by means of adaptive architectures, whereby the e-Learning content is clearly distinguished from the three models that influence the learning process – the user model, the domain model, and the pedagogical model. The user model presents all information related to the learner's individuality, which is essential for the realization of personalized learning process. The domain model presents the structure of the topic/subject for study/learning. In addition, in our architecture we want to support a goal-driven learning, whereby in case of a learner's request sent to the system, a concrete pedagogical goal is generated based on the pedagogical model used. The entire control of the user session afterwards obeys this pedagogical goal. These three models are supported explicitly in our architecture. They represent a strong foundation for seeking opportunities for adaptation to the environmental/context changes so as to ensure more efficient personalized learning (in this sense we aim at realization of a user/domain/pedagogical model-driven optimization).

III. Base Middleware Architecture

The first version of the InfoStation's base middleware architecture, developed accordingly to this approach, is shown in Figure 2. The various agents employed perform different actions as summarized below.

The *Scanner* agent searches for and finds mobile devices (within the range of the InfoStation) with enabled/activated wireless interface corresponding to the type of the InfoStation. In addition, this agent retrieves a list of services required by users (registered on their mobile devices upon installation of the client part of the application and started automatically by the InfoStation agents) and sends this to the *Connection Adviser* agent, which filters the list out. The filtration is carried out with respect to a given (usually heuristic) criterion. Information needed for the filtration is stored in a special database (DB). Finally, the Connection Adviser agent sends the filtered list to the *Connection Initiator* agent, which initiates the communication required for obtaining the service(s) requested by the user. This agent generates the so-called Connection Object, through which a connection with the mobile device is established (e.g. over WiFi, Bluetooth, etc.). In addition, for each active mobile device it generates a corresponding *Connection Agent*, to which it hands over the control of the established wireless connection with this device. The internal architecture of the *Connection Agent* contains three threads: an *Agent Thread* used for communication with the Query Manager agent, and a *Send Thread* and a *Receive Thread*, which look after the wireless communication with the mobile device.

The *Query Manager* agent is one of the most sophisticated components of the InfoStation's architecture. It acts as a data distribution unit between the lower layer and upper layer of the InfoStation. On one hand, the Query Manager prepares and determines where information received from the mobile device is to be directed, e.g. to simple services, or to sophisticated services via interface agents. For this purpose, this agent transforms the messages coming from the Connection Agent into messages of the corresponding protocols, e.g. UDDI or SOAP for simple services. The direct activation of simple services (for example Web services) is possible without the mediation of Interface Agents. The latter are designed to maintain communication with more sophisticated services by using more complex, semantic-oriented protocols, e.g. OWL-S, [22]. In the opposite direction, the Query Manager agent transforms the service execution results into messages understandable by the *Personal Assistant Agent (PAA)* installed on the user mobile device. This is needed because the results must be returned to the relevant PAA, which has requested the provision of the service on behalf of its user.

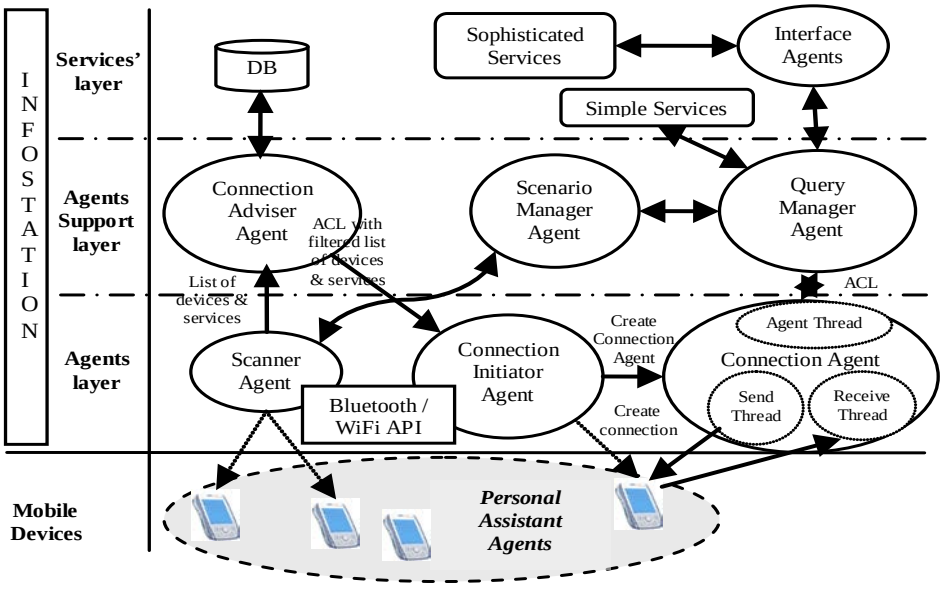

Figure 2. The InfoStation's base middleware architecture

IV. Scenario-based Management

Due to the supported user mobility and device mobility, the following four main communications scenarios are envisaged in the InfoStation architecture [23]:

- '*No change*' – illustrates the straightforward provision of a m-Learning service within the range of the same InfoStation and without change of the user device;
- '*Change of user mobile device*' – due to the inherent user mobility, it is entirely possible that during the m-Learning service provision, the user may shift to another mobile device (e.g. by switching to a device with greater capabilities, the user may experience a much richer service environment and utilize a wider range of resources);
- '*Change of InfoStation*' – within the InfoStation paradigm, the connection between the InfoStations themselves and the user mobile devices is by definition geographically intermittent. With a number of InfoStations positioned around a University campus, the users may pass through a number of InfoStation cells during the service session. This transition between InfoStation cells (i.e. device mobility) must be completely transparent to the user, ensuring the user has apparent un-interrupted access to the service;
- '*Change of InfoStation and user mobile device*' – most complicated scenario whereby the user may change the device simultaneously with the change of the InfoStation.

The scenario management is performed by a dedicated Scenario Manager Agent (SMA) in our middleware architecture. Our aim is to extend and precise further the existent middleware architecture in order to be able to control the service requests' execution independently of the environmental (scenario) changes. Here we will consider the necessary extensions for the control of the device mobility and the corresponding two scenarios - *'No change'* and *'Change of InfoStation'*. First we will present our proposal for the control of the *'No change'* scenario. Then we will show that the '*Change of InfoStation'* scenario could be considered as a synchronization of two *'No change'* scenarios.

For the control of the *'No change'* scenario, a *Request Register* and a *Mobile Device Register* are supported in our architecture. In addition, a relationship is supported between the records in these two registers such that each service request to correspond to a particular mobile device. The Request Register keeps information about all active requests in the system. The assumption is that the execution of a service request starts always according to the *'No change'* scenario. When a request is accomplished and the result returned to the user, the corresponding request registration is deleted from the registry. The Mobile Device Register keeps information about the mobile devices which are currently traced by the middleware within the InfoStation network. When an unregistered mobile device enters the radio range (service area) of an InfoStation, it is first registered. Correspondingly, when the device leaves the InfoStation and the service request initiated by it is already accomplished, the information about it is deleted from the registry. However, if the service request is not accomplished yet, the registration of the corresponding device is not deleted but rather only deactivated. In situations like this, the request execution continues according to the *'No change'* scenario, but the control expects possible changes in the scenario. If in the next moment the mobile device appears within the range of the same InfoStation, the *'No change'* scenario remains unchanged. If the device enters the range of another InfoStation, then the control detects a scenario change and the service execution continues according to a new scenario (i.e. '*Change of InfoStation'* scenario).

As mentioned above, the '*Change of InfoStation'* scenario could be presented as a combination of two *'No change'* scenarios, which, however, are executed in different InfoStations. Moreover, the two scenarios must be synchronized to each other as to figure out whether they are

two independent *'No change'* scenarios or a *'Change of InfoStation'* scenario is taking place (Figure 3).

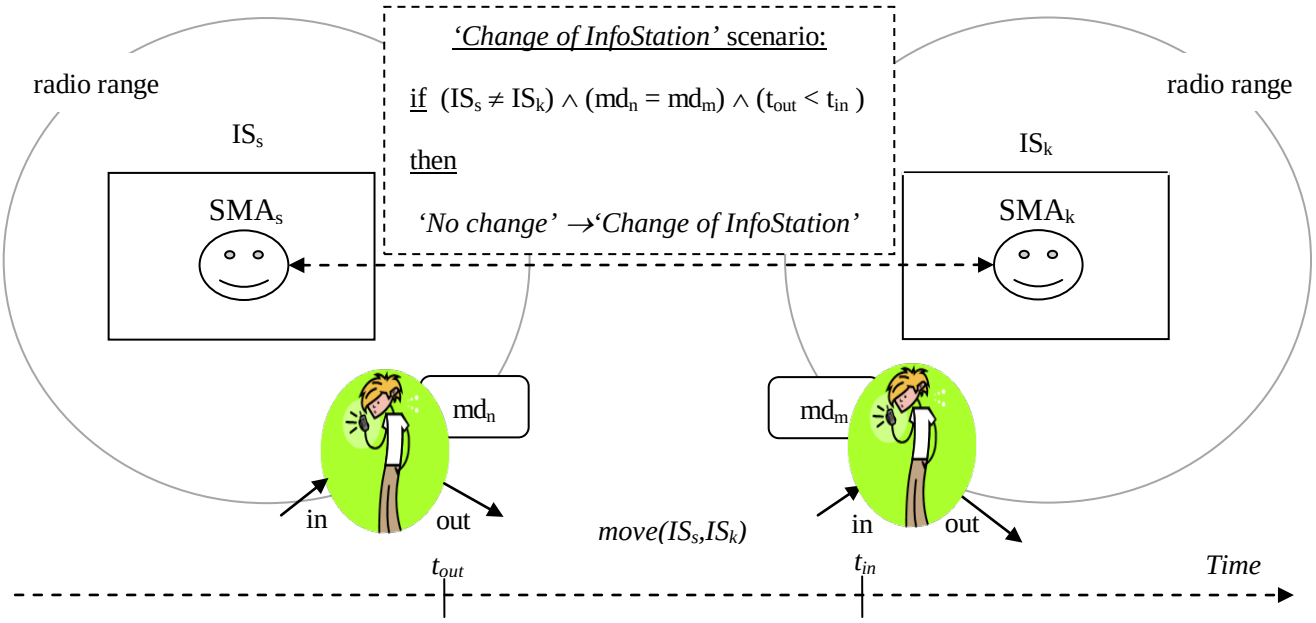

Figure 3. A scenario change during the service request's execution

For the realization of synchronization, we envisage two possible models:

- *InfoStation-Center based Synchronization* – synchronization is done through the InfoStation Center. In this case, the SMA of one InfoStation (SMA_s) sends to the InfoStation Center a Request Register's record of the activated request for mobile device md_n. When md_n enters within the range of another InfoStation IS_k, SMA_k sends the device ID to the InfoStation Center, which returns information specifying if the request is still active and which SMA controls it. Then the two SMA agents agree about the scenario change;
- *Mobile-Device based Synchronization* – in this model, information related to an active service request is kept by the PAA operating on the device. When entering the range of IS_k, the PAA communicates with SMA_k to come to an agreement about the scenario change.

V. Simulation Environment

A simulation environment is being created for testing the presented middleware. By means of this environment the middleware's behavior can be analyzed in the framework of separate experiments. The simulation system is being developed as a wrapper of the tested middleware, in which the following activities are carried out:

- Preparation and initialization of the experiment.
- Activation of the experiment execution.
- Visualization of the middleware's behavior during the experiment execution.
- Automatic generation of protocols, which may be used for the assessment of the middleware's behavior.

The simulation environment consists of two main modules: an *Experiment Organizer (EO)* and an *Experiment Runner (ER)*, Figure 4. EO is a specialized user interface used for the preparation of experiments and presentation of results. Experiments are presented as a specification; the results are recorded and are available for subsequent analysis. Beside this, some results may be visualized. ER is an agent-based wrapper of the tested middleware, which controls the experiment execution/running in accordance with the given specification. The interim results obtained during a particular experiment are transmitted to EO, where the experiment execution can be visualized. A generator of protocols has been developed also in EO. The simulation environment can present all events occurring in the middleware in the form of a log file, e.g. the actual moments when agents/containers/behaviors are created or removed, communications messages exchanged between agents, changes in the state of agents/containers/behaviors, etc.

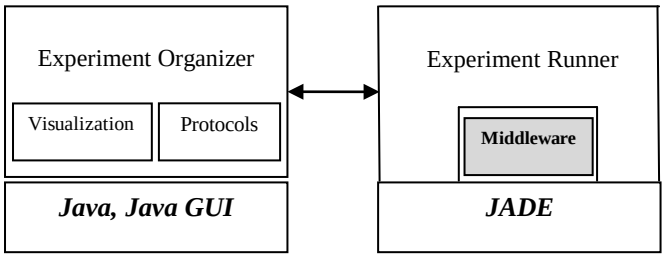

Figure 4. The logical structure of the simulation environment

A. Levels

The experiments, carried out with the help of this environment, allow research and analysis of the middleware's behavior at four levels as described in the following subsections.

1) Scenario level

At this level we want to trace the global behavior of the middleware during the execution of the scenario designated in the experiment. Our particular interest is focused on the behavior of the resident agents and the process of generation of operational (temporary) agents, which depend on the changes occurring in the environment. These changes are specified in the description of the experiment and are simulated during its execution. As a result of the experiment, the following features of the middleware can be analyzed:

- Tracking the movement of users and the corresponding reaction of the middleware.
- Statistics about generated and removed operational agents.
- Starting and completing a communication.
- Data transfer between devices.
- Review of agents on each hardware device.

2) Container level

At this level, for each simulated component (InfoStation Center, InfoStation, or mobile device) the simulation environment shows the active containers living in the middleware. The following information that characterizes a container could be extracted: the address, the identifier (identifying the container in the JADE platform [24]), the logical name, the port, the protocol used for communication, etc. The main container also contains two functional JADE agents: an Agent Management System (AMS) and a Directory Facilitator (DF). AMS is a mandatory component of the platform and is responsible for the overall management of its operation, such as the creation and deletion of agents, overseeing the migration of agents to/from the platform, etc. DF is an optional component providing yellow-pages services to other agents [24].

3) Agent level

At this level the simulation environment shows what agents live in the containers of each InfoStation. The following information characterizing an agent could be extracted: the name of the class whose instance is this particular agent; the local/global name for JADE; the state of the agent; the container in which it is located, etc. Beside this, different characteristics of agents can be visualized, such as the location of the agent (i.e. on a mobile device, InfoStation, or InfoStation Center), creation of an agent (the simulation environment catches the exact creation time and visualizes it along with the location, container, name, state), removal of an agent, etc.

At the agent level a serious consideration is given to the detailed presentation of the states of agents. The agents can be in one of many states, such as *AGENT_STATE_ACTIVE* (the agent is in active state, i.e. at the moment it carries out its functionality and is not awaiting any events from the JADE environment), *AGENT_STATE_DELETED* (the agent is removed, i.e. it has entered into this state just before its removal by the JADE environment and the release of resources held by it), *AGENT_STATE_IDLE* (the agent is in free state, i.e. it has fulfilled the assigned functionality and now it is waiting), *AGENT_STATE_INITIATED* (the agent has been just generated), *AGENT_STATE_SUSPENDED* (the agent is going into inactive "sleep" state), *AGENT_STATE_TRANSIT* (in this state the agent can be moved to another JADE container or another JADE platform), *AGENT_STATE_WAITING* (the agent is going into awaiting state), *AGENT_STATE_COPY* (the agent is going into state, which allows it to be copied), *AGENT_STATE_GONE* (the agent has already moved to another platform or container).

This level collects also information that allows analysis of interactions between agents. The simulation environment may intercept messages exchanged between different agents. The information, which may be visualized for each message, includes: the sender, the recipient, and the contents of the message encoded in ACL.

4) Behavior level

The purpose of this level is to provide opportunities for analysis and assessment of local behavior of individual agents. The agents maintain their own libraries of different "behaviors", which could change depending on the environment. The simulation environment obtains the following information during the process of operation of the observed agents: the name of the behavior, the name of the class whose instance is this behavior, the type of behavior (simple or complex) and other behaviors contained in the current behavior (for complex behaviors only). At this level the simulation environment can visualize the following characteristics of each behavior: the behavior's affiliation to an agent, the creation or addition of behavior, the removal of behavior, etc.

Moreover the simulation environment can distinguish the following states of each behavior: *STATE_READY* (this state shows that the behavior is ready for execution), *STATE_BLOCKED* (the behavior execution has stopped for some reason, e.g. waiting for a message to be received, input/output operation, etc.), *STATE_RUNNING* (the behavior is currently executed).

B. Implementation

This subsection presents in detail the architecture and implementation of the Experiment Runner (ER). ER has an agent-based architecture (Figure 5), implemented by means of JADE, which allows smooth integration of the middleware.

ER consists of the following agents:

- **Simulation Controller** – the desired configuration of the InfoStation (IS) network is simulated with the help of this agent. It is the central coordinator of each experiment that is currently running. This agent issues commands for the creation or removal of simulated InfoStations and mobile platforms, defines waiting intervals and the sequence of interaction between simulated mobile and IS platforms. This agent is also an interpreter to a specialized script language that describes the desired

experiment. Currently we are studying the possibilities for using the Interval Temporal Logics [25] and its interpreter Tempura [26] for describing the experiments.

- **SIS Handler** – this agent controls a separate ***Simulated InfoStation* (SIS) *platform***. Depending on the purpose of the experiment and on the commands received from the Simulation Controller, it generates different commands addressed to the managed SIS and received by the corresponding **Spy agent**. The **SIS Handler** takes care of the creation and removal of the specific SIS, by setting the relevant parameters needed for the operation of the JADE platform and the desirable initialization of the **Spy agent**.
- **Spy Agent** – the role of this agent is to execute the commands issued by the **Simulation Controller** agent on the ***SIS platform***. In addition it provides feedback to the simulation environment on the events that have occurred in the ***SIS platform***.
- **SMD Handler** – this agent controls the simulated mobile device (SMD) in a similar way as the SIS Handler controls the SIS. He interprets the commands received from the **Simulation Controller** agent, processes them and submits corresponding commands to the simulated **PAA**. The SMD Handler takes care of the creation and removal of the ***Simulated Mobile Device*** platform and its desired initialization.
- **Simulated PAA** – this agent plays the role of a user's personal agent; it simulates the user actions based on the commands it receives from the SMD Handler. It also provides feedback by reporting on the events occurring in the simulated mobile device platform and on the exchanged messages. Another task of its own is to connect to the various SIS and to issue requests for execution of services.

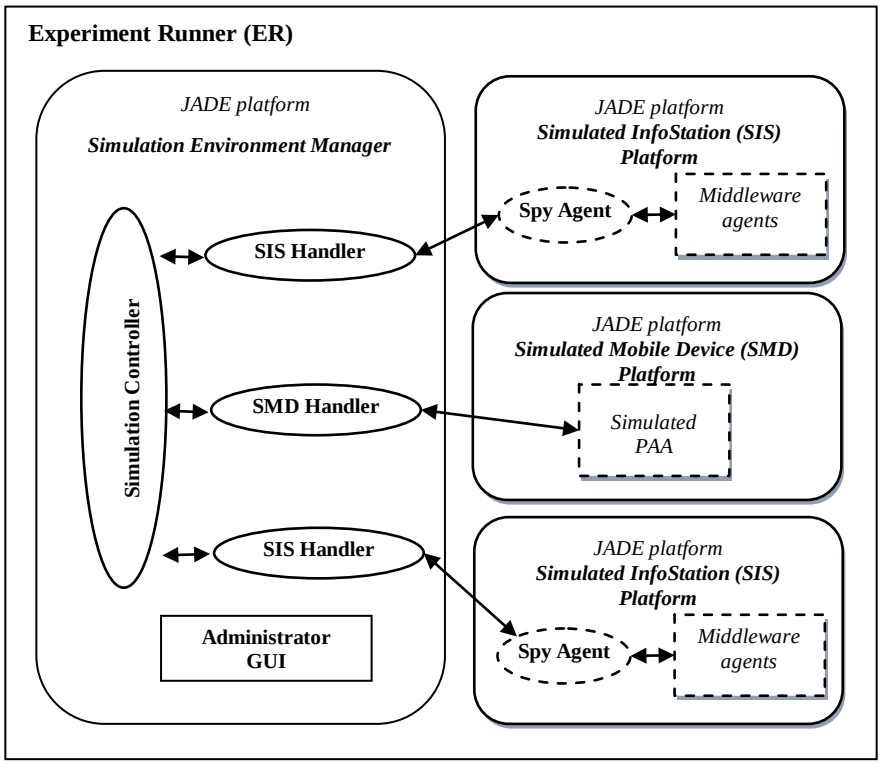

Figure 5. The architecture of the Experiment Runner (ER)

The ER's agents are deployed on the following platforms:

- The **Simulation Environment Manager** is a JADE platform, on which the ***Simulation Controller*** and the ***Handler*** agents live. It serves as a central console for the control of experiments. Work with the environment and visualization of the processes is carried out through a specialized graphical user interface (GUI) used by the administrator of experiments. By using this interface, the administrator of the simulation is able to issue commands to the controller, to start and stop experiments, to monitor the current status of the environment and the front conditions in the form of a log file.
- The **Simulated InfoStation Platform** is a JADE platform on which the unmodified agents of the standard middleware are located. The only addition is the presence of the **Spy agent**, which provides a feedback to the controller. The SISs provide the same conditions for the operation of middleware as the real InfoStations, with only one difference that all of them are deployed on the same computer.
- The **Simulated Mobile Device Platform** is a JADE platform, on which the simulated personal assistant is operating. This platform represents the mobile device of a single user.

C. Spy agents

The AMS agent on each JADE platform offers the possibility to other agents to subscribe for events occurring in the environment. Based on this possibility, in each simulated InfoStation a **Spy agent** is created which to detect the changes in the various JADE platforms and to send relevant information to the corresponding platform handler of the simulation environment. The handler in turn may forward this information to the simulation manager for visualization and further processing.

Spy agents can subscribe to the following events: *ADDED_CONTAINER*, *REMOVED_CONTAINER*, *BORN_AGENT*, *DEAD_AGENT*, *MOVED_AGENT*, *CHANGED_AGENT_STATE*, *ADDED_BEHAVIOR*, *REMOVED_BEHAVIOR*, *CHANGED_BEHAVIOR_STATE*, *SENT_MESSAGE*, *RECEIVED_MESSAGE*, *POSTED_MESSAGE*, *CHANGED_AGENT_STATE*.

In order to be able to receive information about these events, each **Spy agent** must engage in communication with the AMS agent using the ontologies provided by JADE (i.e. the JADE-Agent-Management-ontology, Introspection-ontology, and FIPA-Management-ontology). A **Spy agent** may be configured to provide snooping on a specific group of agents. Information collected by the **Spy agent** could be easily filtered out for subsequent visualization, e.g. to display only that part which is of interest.

VI. CONCLUSION

This paper has presented an OMG's MDA-based approach for the development of a service-oriented and agent-based middleware architecture supporting flexible and adaptable, scenario-based, context-aware provision of m-Learning services within an InfoStation environment. To achieve context-awareness, the middleware is able to identify/detect the changes occurring in the environment and adapt its behavior accordingly. Considering the system development as a process of iterations, our approach provides an extensive ability to examine different development aspects and extend the system architecture step by step. The first two iterations, namely the base middleware architecture and the scenario-based management, have been described in more detail. A simulation environment used for testing the architecture has been also presented.

The presented middleware architecture is implemented by means of the JADE framework [24]. Currently we are looking for a suitable way for formalization of the scenario presentation as to allow the Scenario Manager Agent (SMA) to successfully identify different scenarios executed (or changed) during the run-time. The development of a formal context-awareness model will be based on the Calculus of Context-Aware Ambients (CCA), [27].

Acknowledgment

The authors wish to acknowledge the support of the Bulgarian National Science Fund (Research Projects Ref. No. ДО02-149/2008) and the Telecommunications Research Centre, University of Limerick, Ireland (http://www.ece.ul.ie/trc/).

References

[1] M. O'Droma, I. Ganchev. "The Creation of a Ubiquitous Consumer Wireless World through Strategic ITU-T Standardization", *IEEE Communications Magazine*, 48 (10), pp. 158-165, 2010.

[2] P. Barker. "Designing Teaching Webs: Advantages, Problems and Pitfalls". In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunication, Association for the Advancement of Computing in Education*, pp. 54-59, 2001.

[3] H. Maurer, M. Sapper. "E-Learning Has to be Seen as Part of General Knowledge Management". In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pp. 1249-1253, 2001.

[4] R. H. Frenkiel, T. Imielinski. "InfoStations: The joy of 'many-time, many-where' Communications". *WINLAB Technical Report TR-119*. Rutgers University, New Jersey, USA, 1996.

[5] J-W. Chang, H-J. Lee. "Context-Aware Architecture for Intelligent Application Services in Ubiquitous Computing". In *Proceedings of the International Conference on Semantic Computing*, pp. 275-281, 2007.

[6] E. Goh, et al. "A Context-Aware Architecture for Smart Space Environment". In *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pp. 908-913, 2007.

[7] H. L. Chen. "An Intelligent Broker Architecture for Pervasive Context-Aware Systems". *PhD dissertation*, University of Maryland, USA, 2004.

[8] Z. Qingsheng, et al. "Research on Context-Aware Architecture for Personal Information Privacy Protection", *Cybernetics*, pp. 3912-3916, 2007.

[9] R. Capilla. "Context-aware Architectures for Building Service-Oriented Systems". In *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06)*, pp. 300-303, 2006.

[10] R. Schmohl, U. Baumgarten. "A Generalized Context-aware Architecture in Heterogeneous Mobile Computing Environments". In *Proceedings of the Fourth International Conference on Wireless and Mobile Communications*, pp. 118-124, 2008.

[11] FIPA Agent Communication Language, http://www.fipa.org/repository/aclspecs.html. 2002.

[12] A. K. Dey, G. D. Abowd. "Towards a better understanding of context and context-awareness". In *Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness*, pp. x.1-x.12, 2000.

[13] IMS Abstract Framework: White Paper v1.0, http://www.imsglobal.org/af/afv1p0/imsafwhitepaperv1p0.html. 2003.

[14] S. Stoyanov, et al. "An Approach for the Development of InfoStation-Based eLearning Architectures", *Comptes Rendus de l'Académie bulgare des Sciences*, 61 (9), pp. 1189-1198, 2008.

[15] I. Ganchev, et al. "InfoStation-Based University Campus System Supporting Intelligent Mobile Services", *Journal of Computers*, 2 (3), pp. 21-33. 2007.

[16] I. Ganchev, et al. “Adaptable InfoStation-based mLecture Service Provision within a University Campus”. In *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT’07)*, pp. 165-169, 2007.
[17] OMG Model Driven Architecture, Guide Version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/03-06-01. 2003.
[18] I. Jacobson, et al. *The Unified Software Development Process*. Addison-Wesley. 1999.
[19] I. Ganchev, et al. “InfoStation-Based Adaptable Provision of M-Learning Services: Main Scenarios”, *International Journal "Information Technologies and Knowledge"*, Vol. 2, pp. 475-482, 2008.
[20] I. Ganchev, et al. “InfoStation-based mLearning System Architectures: Some Development Aspects”. In *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies (ICALT’08)*, pp. 504-505, 2008.
[21] S. Stoyanov, et al. “Multi-Agent Architecture for Context-Aware mLearning Provision via InfoStations”. In *Proceedings of the International Workshop on Context-Aware Mobile Learning*, pp. 549-552, 2008.
[22] OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/. 2004.
[23] I. Ganchev, et al. “Communications Scenarios for InfoStation-Based Adaptable Provision of M-Learning Services”. In *Proceedings of the 2nd International Conference “Modern (e-)Learning”*, pp. 98-104, 2007.
[24] JADE 3.7: Java Agent Development Framework, http://jade.tilab.com/. 2009.
[25] B. Moszkowski. *Executing temporal logic programs*. Cambridge University Press, Cambridge, 1986.
[26] Tempura. © STRL 1996-2010. http://www.cse.dmu.ac.uk/STRL/ITL/itlhomepagese6.html.
[27] F. Siewe, et al. “CCA: a Calculus of Context-aware Ambients”. In *Proceedings of 2009 International Conference on Advanced Information Networking and Applications Workshops*, pp. 972-977, 2009.

Author Biographies

Dr Stanimir Stoyanov PhD, MACM, MIEEE was born in Pleven, Bulgaria. He received his bachelor’s and doctoral degrees from the Humboldt University of Berlin. He is a Lecturer and a Head of the Department of Computer Systems, University of Plovdiv, Bulgaria. His research interests include: context-aware and adaptable software architectures, intelligent agents and multi-agent systems, agent- and service-based architectures, middleware, eLearning tools and environments, mobile eLearning services.

Dr Ivan Ganchev is a Senior Member of the Institute of Electrical and Electronic Engineers (IEEE), USA. He received his engineering and doctoral degrees from the Saint-Petersburg State University of Telecommunications in 1989 and 1994, respectively. He is an Associate Professor at the University of Plovdiv and an ITU-T Invited Expert. Currently he is lecturing in the University of Limerick (Ireland), where he also acts as a Deputy Director of the Telecommunications Research Centre. Currently he is a member of two European Science Foundation ‘COoperation in the field of Science and Technology research’ Actions: IC0906 “Wireless Networking for Moving Objects” (WiNeMO) and IC0905 “Techno-Economic Regulatory Framework for Radio Spectrum Access for Cognitive Radio/Software Defined Radio” (TERRA). His research interests include: simulation and modeling of complex telecommunication systems, new communications paradigms for wireless next generation networks (NGN), always best connected & best served (ABC&S), third-party authentication, authorization and accounting (3P-AAA) management, wireless billboard channels (WBC) and cognitive pilot channels (CPC), Internet tomography, mLearning. Dr Ganchev has served on the Technical Program Committees of a number of prestigious international conferences and workshops.

Damyan Mitev was born in Haskovo, Bulgaria. He received his bachelor’s and master’s degrees from the University of Plovdiv “Paisii Hilendarski”. He is a final-year PhD student and an assistant at the Department of Computer Systems, University of Plovdiv. His interests include: multi-agent architectures, eLearning tools, dynamic behavior generation, software engineering and reengineering tools and techniques.

Vladimir Valkanov was born in Plovdiv, Bulgaria. He received his bachelor’s and master’s degrees from the University of Plovdiv “Paisii Hilendarski”. He is a final-year PhD student at the Bulgarian Academy of Sciences and an assistant at the Department of Computer Systems, University of Plovdiv. His interests include: intelligent agents and multi-agents architectures, middleware, eLearning tools, software engineering and reengineering tools and techniques.

Dr Máirtín O’Droma received his BE and PhD degrees from the National University of Ireland in 1973 and 1978, respectively. He is a Senior Academic and Director of the Telecommunications Research Centre at the University of Limerick, Ireland. He is an IEEE Subject Matter Expert, an ITU-T Invited Expert, a Fellow of the IET and Chair of Ireland's Royal Irish Academy's Communications and Radio Science Committee. Currently he is delegate (Ireland) to the COST Actions IC0906 “Wireless Networking for Moving Objects” (WiNeMO) and IC0905 “Techno-Economic Regulatory Framework for Radio Spectrum Access for Cognitive Radio/Software Defined Radio” (TERRA). His research interests include: wireless NGN infrastructural innovations and new paradigms; complex wireless telecommunication systems simulation and behavioral modeling, linearization & efficiency techniques in multimode, multicarrier broadband nonlinear microwave and mm-wave transmit power amplifiers; smart antenna and MIMO channels, mLearning. Dr O'Droma has served on the Technical Program Committees of numerous IEEE and other international conferences and workshops.