Agent-Based Mobile Event Notification System

doi:10.3991/ijim.v4i4.1427

R.F. El-Gazzar, O. Badawy and M. Kholief
Arab Academy for Science and Technology and Maritime Transport, Alexandria, Egypt

***Abstract*—In recent years, the noticeable move towards using mobile devices (mobile phones and PDAs) and wireless technologies have made information available in the context of "anytime, anywhere using any mobile device" experience. Delivering information to mobile devices needs some sort of communication means such as Push, Pull, or mixed (Push and Pull) technologies to deliver any chunk of information (events, ads, advisory tips, learning materials, etc.). Events are the most important pieces of information that should be delivered timely wherever the user is. Agent-based technology offers autonomous, flexible, adaptable, and reliable way of delivering events to any device, anywhere, and on time. Publish/subscribe communication model is the basic infrastructure for event-based communication. In this paper, we define the need to mobilize the event notification process in educational environment and the possible categories of event notifications that students can receive from their educational institution. This paper also proposes a framework for agent-based mobile event notification system. The proposed framework is derived from the concept of push–based publish/subscribe communication model but taking advantage from software agents to serve in the mobile environment. Finally, the paper provides a detailed analysis for the proposed system.**

***Index Terms*—Mobile systems, agent-based systems, event notification systems, publish/subscribe model.**

I. INTRODUCTION

Mobile devices are becoming popular in use nowadays as mobile computing and supportive mobile telecommunications network infrastructure (GSM, GPRS, UMTS, EDGE, Wi-Fi, and WiMAX) have emerged and rapidly evolve. A mobile device can be any device that is small, autonomous, and unobtrusive enough to be carried in everyday life with the user such as: (1) PDAs/smart phones (blackberry, iPhone). (2) Digital phones (Nokia, Sony Ericsson). (3) Non-telephony devices (Apple's iPod). (4) Mobile computers (tablet PC, NoteBook, and Laptop) [5]. However, the mostly carried mobile devices with users nowadays are mobile phones and PDAs. With mobile devices, users can have useful piece of information while on bus, on train, at home, or having their leisure time somewhere.

Pull communication requires users to initiate requests for information. This causes network overload, but it addresses privacy concerns associated with the Push approach [12]. Push communication involves initiating the delivery of needed information based on the users' predefined preferences. Thus the load on network is reduced [10]. Hence, Push technology is a great choice for delivering important events to users in a timely fashion.

Events are very important pieces of information that need to be known in the right time in order to be valuable, because if such information is delivered late, it will be meaningless. Events can be categorized into:

1. **User-interest events:** are related to the user's predefined interests. In the case of university, the user-interest events could be trips, conferences, competitions, etc.
2. **User-related events:** are related to but not predefined by the user; these events are often very important and identified by the institution or the community the user belongs to. In the case of university, the user-related events could be registration date, exams schedule, course notifications, etc.

Besides timely Push service, events should be delivered according to users' interests. Event-based communication is mostly based on the notion of publish/subscribe model through which publishers publish events; these events are pushed to the subscribers according to their interests. Such advantages need an intelligent and reliable approach; hence, agent-based technology is a good choice. Agent-based technology offers autonomous, flexible, adaptable, and reliable way of delivering events to any device, anywhere, and on time.

Software agents are intelligent software programs that perform certain tasks on the user's behalf in autonomous, reactive, proactive and adaptive behavior [4]. Thus agents know what to do, how to do it, and when to do it; this promotes high level of autonomy, reactiveness, and proactiveness [16]. The Foundation for Intelligent Physical Agents (FIPA) provides standards for message transport protocols, Agent Communication Language (ACL), content languages, and interaction protocols for the sake of interoperability [13].

Taking benefits from the four technologies mentioned above (mobile computing, Push technology, software agents, and publish/subscribe model) this research proposes a framework for an agent-based mobile event notification system. Section 2 discusses a literature review on three advanced technologies used in the proposed framework. Section 3 describes the requirements for establishing event notification system in a mobile environment. Section 4 defines the problem to be solved in this paper. Section 5 describes the proposed framework architecture and detailed analysis of the system. Section 6 provides conclusions on the framework.

II. LITERATURE REVIEW

As a result of inventing more advanced mobile devices and the rapid evolution in wireless network infrastructure [1], [15], the use of mobile devices started to be beyond voice calls, video calls, text messaging (SMS), and multimedia messaging (MMS) activities. Nowadays, mobile devices are being used in many important activities in many fields in our life such as m-learning, m-banking, and m-advertising [1], [7]. Such mobile activities are served according to the user's preferences, location, and device limitation. Hence, the mobile computing promotes flexibility, mobility, and adaptability through small, light, and movable devices.

For establishing event-based communication, the most popular push model is publish/subscribe communication model. The motive for using such paradigm arises from the need for an asynchronous, loosely coupled, and many-to-many communication in the context of mobile and/or large-scale distributed systems. Such model enables subscribers to express their interests in event notifications; these events are produced by publishers and delivered by event service to subscribers only if events match their interests. In general, publish/subscribe model consists of publishers, subscribers, and dispatch or event service as shown in Fig. 1. Subscribers issue subscriptions to express their interests in events by calling subscribe() operation on the event service without knowing the sources of these events. Subscribers can terminate their subscriptions by calling unsubscribe() operation. Publishers produce events by calling publish() operation without knowing the identity of subscribers who will be notified of that event. Event service provides storage and management of subscriptions, storage and filtering events, and efficient delivery of event notifications to interested subscribers by calling a set of operations such as updatesubacription(), store_event(), perform_matching(), and route_event() while event delivery could be through e-mail, SMS, or WAP messages. An event is asynchronously propagated to all subscribers who are interested in that event. Hence, event service serves as a neutral mediator between publishers and subscribers. Publishers are producers of events and subscribers are consumers of events. The event is delivered through route_event() operation that invokes remotely the notify() operation at the subscriber end.

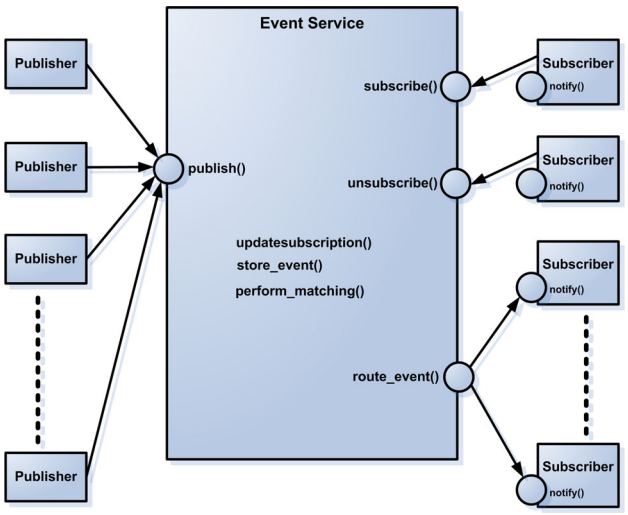

Fig. 1: Components of Publish/Subscribe Model

Hence, event service is responsible for four main operations:

1. **Managing subscriptions:** assuming that updatesubscription() operation is responsible for managing subscriptions, by calling this operation, event service can add, update, and delete subscriptions. Those subscriptions are stored for later filtering purposes. A single subscription contains subscriber's profile and the selected interesting events.
2. **Managing incoming events:** besides managing subscriptions, event service handles the incoming events from the publishers. These events are received in the form of messages; each event message contains event header and body. Events are stored using store_event() operation and used for filtering purposes.
3. **Filtering events:** event service calls perform_matching() operation that is responsible for filtering events. Filtering involves matching incoming events to interested subscribers using various filtering techniques such as fuzzy-logic, rule-based, semantics, etc.
4. **Delivering events:** after filtering, event service invokes route_Event() operation to deliver events to interested subscribers in a timely fashion through an appropriate communication channel.

The mantra for event-based publish/subscribe systems is the "decoupling" between publishers and subscribers. Decoupling is the key characteristic that distinguishes publish/subscribe communication from the other alternate communication paradigms such as message passing, RPC/RMI, notifications, shared spaces, and message queuing. Such paradigms have proved their inability to support fully decoupled communication between publishers and subscribers [11]. Decoupling can be decomposed into four dimensions:

1. **Space decoupling:** sometimes called "anonymity" where publishers do not need to know the address and identity of subscribers and subscribers do not need to know the identity of publishers as well.
2. **Time decoupling:** both publishers and subscribers do not need to exist at the same time; publishers can publish events while subscribers are disconnected and subscribers get notified of an event while the publisher of that event is disconnected.
3. **Synchronization decoupling:** where subscribers receive event notifications while being not connected to the system or doing other concurrent activities. The production and consumption of events is decoupled from the flow of control of the publishers and subscribers. Hence, publishers publish events without waiting for results and subscribers receive events without explicitly waiting for these events, thus promoting flexibility.
4. **Data decoupling:** in which subscribers only receive data that they are interested in, and the event service may modify data if needed. This

avoids delivering uninteresting events to subscribers and thus saves resources that would process these uninteresting events.

Software agents have been integrated with publish/subscribe model to come up with "Rendezvous-Notify" framework that serves mobility requirements of the mobile environment [13]. This framework is able to maintain disconnected operations and manage subscriptions efficiently. Rendezvous-Notify framework involves using event service agents that are responsible for maintaining subscriptions, buffering events, managing event channels, and routing events.

Mobile agents have been used in the applications of distributed events systems based on publish/subscribe communication protocol [11]. This approach involves using mobile agents as mediators between publishers and subscribers of events. In such scenario, subscribers are required to register in the system and define the events they are interested in, and publishers create mobile agents with the event to be published and dispatch them to the event server in which the mobile agents find out the interested subscribers to push the event to them. Thus flexibility is promoted via asynchronous communication.

Publish/Subscribe model has contributed to delivering information in the mobile context. Hence, a publish/subscribe middleware was proposed to address the requirements of mobile computing applications [3], this middleware provides asynchronous communication as the publishers needn't to have direct contact with the subscribers, thus wireless connection failure is not a problem anymore. Also many-to-many decoupled interaction is provided as many publishers publish events and these events are sent to many subscribers without publishers being connected to the system. Anonymity is supported by such a model as publishers do not have to know the identity of the subscribers and vice versa. Implicit determination of the event notification receivers is provided rather than choosing them by the publishers. Consequently, the system is capable of dealing with a large number of mobile users. However, mobile publish/subscribe applications have been classified into two categories: (1) Static applications that reside in the user's mobile device, as the user is moving, the application can access the system network from different access points. (2) Mobile agent based applications that are able to execute autonomously on any host device to access the system. The first category has been implemented to support mobility service using client proxy [2]. This mobility proxy is an interface medium between the client and the publish/subscribe system while being in disconnected mode.

III. EVENT NOTIFICATION IN MOBILE ENVIRONMENT

As we deal in our research with mobile environment, the possible delivery methods for event notifications are SMSs or WAP messages. Additionally, the mobile pub/sub system should offer a set of benefits such as: (1) timeliness that is achieved by pushing data to interested subscribers once it's produced by the publisher. (2) Asynchronous communication that enables delivering notifications while the subscriber is not connected to the system, thus reliability is guaranteed. (3) Anonymous communication where publishers do not need to know the identity of subscribers, thus flexibility is insured. (4) Supporting logical mobility (the user can receive notifications even if changed her/his mobile device) and physical mobility (the user can be notified anywhere). (5) Expressiveness that is the ability of event service to well-define interests of subscribers. (6) Implicit matching where the event service determines the target mobile subscribers who will receive notifications based on their subscriptions without needing publishers to choose recipients. (7) The ability to manage a large number of potential mobile subscribers allowing for manipulation of their subscriptions (update, insert, and delete). (8) The ability to manage a large number of publishers. (9) Support for simultaneous delivery of notifications to thousands of mobile subscribers. (10) Robustness guarantees delivery of notifications to all target mobile subscribers even in case of network failure while subscribers are moving (that is a characteristic of mobile network) by resending notifications to subscribers who could not be reached previously [3]. The common standard technologies for establishing mobile event notification systems are Common Object Request Broker Architecture (CORBA), Java Message Service (JMS), and Wireless Message Transport Protocol (WMTP) [9].

IV. EVENT NOTIFICATION IN EDUCATIONAL ENVIRONMENT

Educational environment, especially university, is filled-up with various activities and events that are offered for students. These events are usually announced orally or posted paperly on the pin board. The problem is that students have different class schedules, so they exist in different time frames. Within these different time frames, some events might be announced, started, and finished without the student being notified. Also, students may forget to check the pin board due to their busy day schedule. Further problematic case, if a lecturer, for urgent matter, needs to cancel a lecture in the same lecture day, then the oral and pin board methods will be useless. Consequently, there is no option to notify students early and they will only know very late. However, events in university context can be classified into:

- **Social events:** include notifications about incoming trips, sport competitions, conferences, and symposiums.
- **Career events:** include notifications about incoming job opportunities, internships, and scholarships.
- **Academic events:** comprise of registration events (include notifications about registration date, list of available courses for the new semester, timetable of the registered courses) and course notifications (include notifications about announcement for new lecture, canceling lecture, and exams schedule).
- **Warnings:** include notifications to students who exceeded the allowed absence rate.
- **Library events:** include announcements about new available resources, acknowledging student that the requested resources are sent to her/his e-mail, and notifying student to renew their membership.

University should be active by choosing more flexible method to convey the previously listed events to students

anywhere and timely. Hence, mobile event notification system is the best choice. The remaining sections of this paper issue a detailed explanation about the proposed framework for agent based mobile event notification system, scenario analysis and a discussion to analyze how it meets requirements of the mobile environment.

V. PROPOSED FRAMEWORK FOR AGENT-BASED MOBILE EVENT NOTIFICATION SYSTEM

The proposed ABMENS framework is an agent based mobile event notification system that benefits from using software agents to push interesting events to the target students on their mobile devices. The framework embodies the idea of publish/subscribe (pub/sub) Push communication model but in a mobile context. The ideal pub/sub model consists of: (1) Publishers who publish events, (2) Subscribers who express their interesting events by issuing subscriptions, and (3) Event service that manages subscriptions and delivers events to subscribers. The proposed framework as shown in Fig. 2 consists of six main components:

1. **Publisher:** who produces the event data; the publisher could be any member of university staff.
2. **Event Agent (EA):** is a light-weight agent resides in the event provider's mobile device. EA serves as an interface agent enabling event provider to enter events.
3. **Subscriber:** who is a student subscribed for events once she/he is admitted and registered in university for the first time.
4. **Personal Agent (PA):** is a light-weight agent resides in the student's mobile device. PA is identified by the student ID. PA serves as an interface agent enabling student to select interesting events, also PA displays event notifications to student.
5. **Agent-Based Mobile Event Notification System (ABMENS):** in the role of event service that serves as a middleware between university database server layer and both event providers and students. The ABMENS receives events from staff through EA, stores events and matches them with interested students, and delivers event notifications to the target students through their PAs. Also the ABMENS allows for manipulating and storing students' subscriptions. The ABMENS performs all these functions through a set of four autonomous software agents: (1) Detector Agent (DA) that receives event data from the EA, gives a copy of event data to the LA, and sends the event data to the DB A to be stored into database. (2) Logic Agent receives event data from the DA, retrieves the list of interested students from the DB A, and sends [event data + list of students] to the MA. (3) Manager Agent (MA) receives [event data + list of students] from the LA and deliver event notifications to the target students' PAs, also MA receives subscriptions from the PAs and sends them to the DB A to be stored into database. (4) Trustee Agent (TA) monitors activities of all agents in the ABMENS and ensures synchronization between those agents.

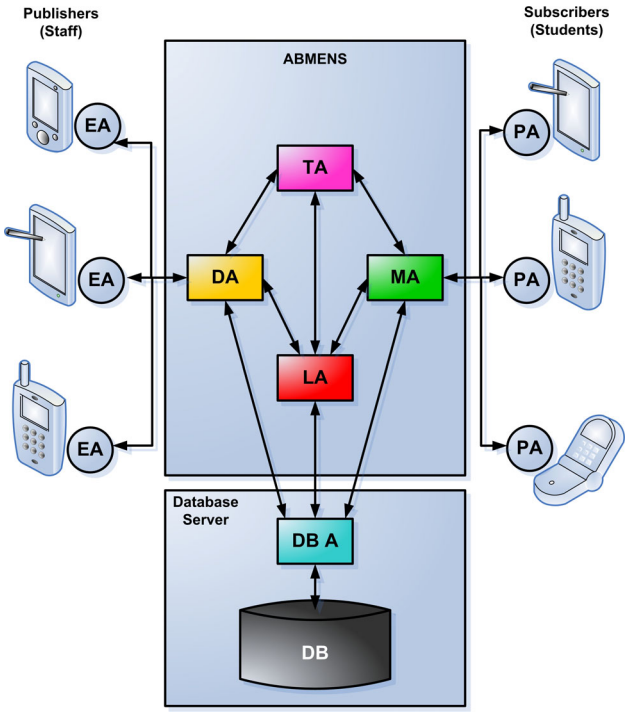

Fig. 2: ABMENS System Architecture

6. **Database Server:** contains the database tables for students, courses, and events and so on. The incoming queries from the ABMENS agents are handled by Database Agent (DB A) that resides in the database server. The DB A receives the following queries from the ABMENS agents: (1) INSERT and SELECT event from the DA. (2) SELECT target students from the LA. (3) INSERT and UPDATE subscription from the MA.

In wireless networks, disconnection and failure to deliver notifications is possible. In such cases, the MA is responsible for resending the notifications to the PAs that could not reach before and ensuring that all target students' PAs have received notifications.

Notice that some events are produced by the event providers (e.g. social events and career events) and some others are propagated by the ABMENS (e.g. warnings and available courses for new semester). The later ones are the responsibility of the DA; the DA detects any new events stored in the database by other subsystems (e.g. grading system and registration system) and retrieves them through the DB A, then sends them to the LA. However, those propagated events are outside the scope of this paper. Therefore, there are two types of subscriptions: (1) Direct subscription: in which students subscribe themselves to events like social and career events. (2) Indirect subscription: in which students receive notification with out issuing subscription like in course notification case. The typical scenario for the proposed system is composed into:

1. **Subscription scenario:** the involved agents are the PA, MA, and DB A. The scenario starts when the student uses the PA to select the interesting event categories, and then the PA sends the selected events to the MA that will send them to the DB A that stores

them into database. Same scenario is applied for unsubscription or updating subscription.

2. **Event notification scenario:** starts when the staff uses the EA to publish an event by entering event data (event category, event title, event date, event place, and description). The EA sends event data to the DA that will send a copy of event data to the LA, also the DA will send event data to the DB A to store it in database. After receiving event copy from the DA, then LA sends SELECT query to the DB A to retrieve list of students' IDs who are interested in that event. The LA receives the query result (list of IDs of subscribed students) from the DB A, then attaches the event data with that list and sends them to the MA that will disseminate the event notifications to PAs of the target students. The PA displays the notification to the student in friendly look.

Assuming that students already issued subscription, the whole flow of activities through the system is depicted in Fig. 3, starting from the staff that enters the event, each agent's activities on the event, ending with the delivery of event notifications to the target students. Note that the PA is identified by student's ID as a global identifier. When the staff enters an event, Event Agent collects event details that consist of: Event Category (social, career, academic, or library events), Event Title, Event Date/Time, Event Place (optional), and Event Description.

Regarding the two following queries involved in the matching process:

Q1:

```
SELECT StudentEvent.Student_ID
FROM Student INNER JOIN (Event INNER
JOIN StudentEvent ON Event.Event_ID =
StudentEvent.Event_ID) ON
Student.Student_ID = StudentEvent.Student_ID
WHERE Event_Category = 'Social';
```

Q2:

```
SELECT Registration.Student_ID
FROM Student INNER JOIN (Course INNER
JOIN Registration ON Course.Course_Code =
Registration.Course_Code) ON
Student.Student_ID = Registration.Student_ID
WHERE Course.Course_Name = 'MIS';
```

Logic Agent (LA) receives event copy from the DA, and based on the event category, it matches the event to the corresponding students, and prepares the appropriate query to send to Database Agent. Thus the following logic is executed:

- IF event category is social or career, THEN prepare a query to retrieve students' IDs who are subscribed to that category of event, and send it to Database Agent.
- IF event category is course notification, THEN prepare a query to retrieve students' IDs who are enrolled in that course, and send it to Database Agent.
- After receiving query result from the Database Agent, the LA attaches the event with the list of IDs of students who will receive the notifications, and sends them to the Manager Agent.

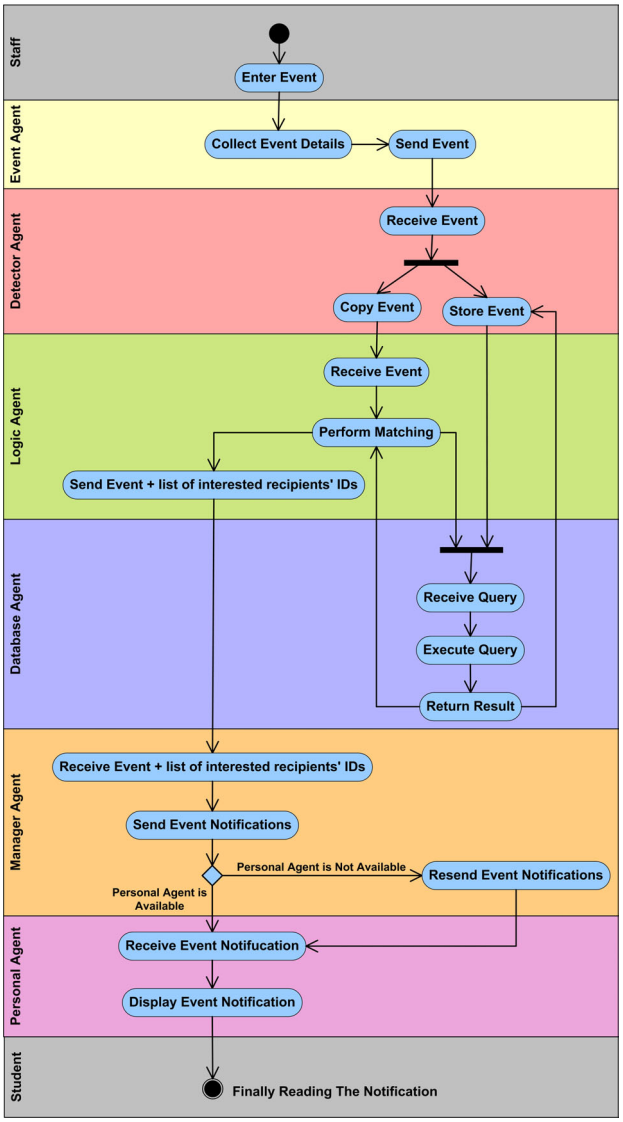

Fig. 3: System Workflow

Manager Agent looks for Personal Agents (PAs) whose IDs match IDs of recipient students using the Directory Facilitator (DF), then sends event notification to these PAs.

Generally, typical publish/subscribe model consists of two basic models (subscription model and publication model) and two basic mechanisms (matching and routing). Publication model defines the data model for publishable event data; hence, events could be structured in various forms (simple unstructured strings, record type, class type, or XML document). Subscription model defines the selection constraints on the published events. Subscription model can be implemented in many forms (relational model, rule definition language, XPath, object-oriented languages, or fuzzy logic rules). Matching could be performed by either XML queries or SQL queries. Routing could be done by flooding, selective routing or gossiping.

Accordingly, the proposed framework applies class type structure to event publication model, relational approach to subscription model, SQL queries to matching events to corresponding subscribers, an agent discovery for routing events to the matching subscribers' agents.

Agent discovery mechanism is provided by the agent platform along with agent registration service.

VI. CONCLUSION

The paper proposed a framework for agent-based mobile event notification system to deliver events offered by university to students on their mobile devices. The paper provided an analysis of the system scenarios and functionalities. The system should promote flexibility, intelligence, and reliability. The system helps university to reach students and keep them updated with the incoming and even the urgent sudden events. The system is currently under development based on the analysis produced in this paper. The chosen platform for design and implementation is the JADE framework because it enables developing light-weight agents using its extended library JADE-LEAP [6].

REFERENCES

[1] A. Adi, Z. Denfgyin, L. Haibo, "M-learning in review: Technology, standard and evaluation," *Journal of Communication and Computer, USA*, Vol. 5, No. 11, pp. 1-6, Nov. 2008.

[2] M. Caporuscio, A. Carzaniga, A. Wolf, "Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications", *IEEE Transactions on Software Engineering*, Vol. 29, No. 12, pp. 1059-1071, Dec. 2003. doi:10.1109/TSE.2003.1265521

[3] G. Cugola, H. Jacobsen, "Using Publish/Subscribe Middleware for Mobile Systems," *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 6, No. 4, pp. 25-33, Oct. 2002. doi:10.1145/643550.643552

[4] Erlin, Y. Norazah, A. R. Azizah, "Overview on Agent Application to Support Collaborative Learning Interaction," *US-China Education Review*, Vol. 5, No. 1, Jan. 2008.

[5] Kineo and UFI/Learndirect (2009). "Mobile Learning Reviewed," Retrieved from http://www.kineo.co.uk/documents/Mobile_learning_reviewed_final.pdf [Accessed June 2, 2009].

[6] A. Moreno, A. Valls, A. Viejo, "Using JADE-LEAP to implement agents in mobile devices," *TILAB "EXP in search of innovation"*, 2003, Italy, http://jade.tilab.com/papers-exp.htm

[7] F. Mousumi, S. Jamil, "Push Pull Services Offering SMS Based m-Banking System in Context of Bangladesh," *International Arab Journal of e-Technology*, Vol. 1, No. 3, pp. 79-88, January 2010.

[8] H. Nwana, M. Wooldridge, "Software Agent Technologies," Intelligent Systems Research, Advanced Applications and Technology, BT technology journal, 1996.

[9] P. Pietzuch, G, Muhl, L. Fiege, "Distributed Event-Based Systems: An Emerging Community," *IEEE Distributed Systems Online*, Vol. 8, No. 2, Feb. 2007. doi:10.1109/MDSO.2007.8

[10] P. M. Reddy, "Mobile Agents: Intelligent Assistants on the Internet," Resonance**:** a journal of science education, published by Indian Academy of Sciences, Bangalore, India, July 2002.

[11] O. K. Sahingoz, N. Erdogan, "Agvent: an agent based distributed event system," Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.112.431 [Accessed April 11, 2010].

[12] X. Shen, F. Li, "The Application of Agent-Based Information Push Technology in Mobile Learning," *International Conference on Information Technology and Computer Science*, vol. 2, pp.192-195, July 2009. doi:10.1109/ITCS.2009.178

[13] S. Tarkoma, "Distributed Event Dissemination for Ubiquitous Agents," *The 10th ISPE International Conference on Concurrent Engineering (CE-2003)*, pp. 105-110, 2003.

[14] R. Unni, R. Harmon, "Perceived effectiveness of Push vs. Pull mobile location-based advertising," *Journal of Interactive Advertising (Spring 2007)*, Vol. 7, No. 2, pp. 28-40, 2007.

[15] A. Vochin, "History of Mobile Phones," Retrieved from http://gadgets.softpedia.com/newsPDF/History-of-Mobile-Phones-3578.pdf [Accessed April 10, 2010].

[16] M. Wooldridge, "Agent-based computing," Interoperable Communication Networks, Vol. 1, No. 1, pp. 71-97, January 1998.

AUTHORS

R. F. El-Gazzar is master student in information systems with the Arab Academy for Science and Technology and Maritime Transport. 1029 Alexandria, Egypt (e-mail: rania.elgazzar@gmail.com).

O. Badawy, Prof. Dr., is with the Arab Academy for Science and Technology and Maritime Transport. 1029 Alexandria, Egypt (e-mail: obadawy@aast.edu).

M. Kholief is with the Arab Academy for Science and Technology and Maritime Transport. 1029 Alexandria, Egypt (e-mail: kholief@aast.edu).

Manuscript received 16 August 2010. Published as submitted by the authors September 27, 2010.