Agent Technologies for M-Services

Zakaria Maamar

Software Agents Research Group$_{@zu}$
College of Information Systems
Zayed University
PO Box 19282, Dubai
United Arab Emirates
zakaria.maamar@zu.ac.ae

Abstract The purpose of the **S**oftware **A**gents for **MO**bile **S**ervices (SAMOS) project is to understand the role of m-services in the wireless world. M-services are an extension to the paradigm of Web/e-services. The wireless world has its own features that make it completely different from the wired world. For instance, new communication channels need to be deployed, new user-friendly services need to be suggested, and new types of computing resources need to be involved. In addition to the traditional challenges of mobile computing, additional issues are addressed in SAMOS such as discovering m-services in an open wireless environment and composing m-services into sequences.
Keywords: software agents, m-services, wireless devices.

1 Overview

According to [2], the Internet is becoming a vehicle of services rather than just a static repository of information. Currently, several businesses are making their services accessible over the web. Usually, Web services or e-services denote these services; flight booking is a typical example. [10] defines an e-service as any asset that is made available via the Internet to drive new revenue streams or create new efficiencies. Besides the new role of the Internet, we expect that more e-services will be offered to people who use wireless devices to conduct their daily operations. We, also, expect that wireless devices will be enhanced with new advanced computing resources [12]. We all witness the growth in the development and use of wireless devices across the world. Unfortunately, this growth does not come alone; it is accompanied with its lot of problems and challenges. For example, wireless devices are strictly bound to their batteries for operation leading to limit the computation power within these devices [9]. In order to support wireless devices-oriented users, we aim at working in SAMOS (**S**oftware **A**gents for **MO**bile **S**ervices) on a new generation of e-services that are specially designed and developed to satisfy the requirements of wireless devices. M-services denote this new generation of e-services.

There exist several types of wireless devices, varying from mobile phones to Personal Digital Assistants (PDAs). These devices present several shortcomings

that make their use requires specific arrangements. Among these arrangements, the services to be offered have to be tailored to each wireless device individually. Unfortunately, this contradicts the platform independence principle of designing e-services. Thus, leveraging e-services into m-services becomes a necessity. M-services will definitely have to consider the characteristics of the wireless devices on which they will be running. In fact, m-services will be fetched on-demand from provider sites to users' wireless devices for execution.

Having pre-installed services on users' wireless devices is an option that can be considered. However, this option would have worked well for a small number of users where looking for the lowest common denominator between them is feasible. In an open environment with multiple users and multiple needs, looking for the common denominator is not obvious. Therefore, on-demand delivery of services to wireless devices is more appropriate than pre-installing services. We advocate that enabling the downloading of dynamically composed services is the approach to follow for systems designed for the wireless world.

In this paper, the following aspects are discussed according to the principles and features of the design of SAMOS. How to assist users with SAs? How to deal with the characteristics of wireless devices, for example limited resources? How to offload computing from these devices to other platforms? How to use interaction diagrams and communication languages to specify the operating of wireless system, such as SAMOS? And, how to combine m-services into sequences? [13] states that many mobile-device users are eager and ready to pay for advanced features, like wireless web access and location-oriented services.

Section *2* motivates the SAMOS project. Section *3* introduces e-services *vs.* m-services. Section *4* presents the SAMOS environment in terms of architecture, types of software agents, and operating. Section *5* presents related work to SAMOS. Finally, Section *6* concludes the paper.

2 Motivating scenario

To motivate our research in SAMOS, we consider the following "fictive" scenario. It consists of a sales representative who works for a healthcare equipments company. The company headquarter is located in country X but serves customers from around the world. Since the representative is most of the time on the move visiting customers, he is equipped with a PDA. It may happen that in preparation for an important meeting with a customer, the sales representative has to carry out certain operations such as simulating the outcomes of the meeting on the company stock shares for the next six months. Unfortunately, before he leaves his office this time the representative did not download the appropriate software applications into his PDA. Thus, he decides to proceed as follows. First, he sends wirelessly a request to the server. According to the workload of the server, one of the following situations occurs:

1. Server "not-busy": the server accepts the request and processes it. To this end, the representative has to send wirelessly the required data. Later on, the server transmits wirelessly the results to the representative's PDA. If the

representative has known before that the server was not busy, he could have attached the required data to the request he submitted in the first time. Unfortunately, this is difficult to predict.

2. Server "busy": the server suggests two alternatives to the representative. The first one is to put his request in a waiting list. The second one is to send the required applications to his PDA for execution. Submitting these applications has to follow specific rules, as it will be discussed in this paper.

In our work, we are interested in situation *Server "busy"* - second alternative, where a wireless device will be acting as a computing platform to the applications to be received from the server. We consider these applications as m-services.

3 E-services *vs.* M-services

E-services - The rationale of creating e-services is to make them accessible anytime, anywhere, and for anyone [4]. An e-service is an application component that an organization provides in order to be assembled and re-used in a distributed, Internet-based environment [9]. An application component is considered as an e-service if it is: 1) independent as much as possible from specific platforms and computing paradigms; 2) developed mainly for inter-organizational situations rather than for intra-organizational situations; and 3) easily composable; its assembling with other e-services does not require the development of complex adapters.

M-services - It may happen that users have to postpone their operations because they lack appropriate facilities running on their devices. Therefore, it becomes important to support such users by allowing them: 1) to search for additional facilities, when needed; 2) to fetch these facilities to their wireless devices; and 3) to conduct the aforementioned operations in a transparent way. In SAMOS, different solutions are put forward. A solution to point 1) consists of devising brokering mechanisms. A solution to point 2) consists of using wireless communication channels. Finally, a solution to point 3) consists of using Software Agents (SAs) to make searching for facilities and fetching them transparent to users [5]. In SAMOS, we consider a component as an m-service if it is: 1) transportable through wireless networks; 2) flexible in term of composition with other m-services; 3) adaptable according to the computing characteristics of the wireless devices; and 4) last but not least, runnable on wireless devices.

4 SAMOS Environment

4.1 Architecture

In the previous section, brokering mechanisms and SAs are suggested as potential candidates to the design and development of a system offering m-services to wireless devices-oriented users. In a nutshell, the features of the SAMOS architecture are:

– Develop three types of SAs: user-agent, provider-agent, and device-agent. The first type is associated with users of m-services (i.e. consumers). The second and third type is associated with providers of m-services.
– Create a Meeting Infrastructure (MI) to be headed by a supervisor-agent [6].
– Develop two types of delegates: user-delegate and provider-delegate. Delegates interact respectively on behalf of user-agents and provider-agents within the MI.
– Develop storage servers that are used for saving the sequence of m-services to be sent to wireless devices for execution. These servers are spread across networks and storage-agents are responsible of their management.

Figure 1 illustrates the architecture of the SAMOS system. It consists of four parts: user, provider, MI, and storage. The MI and storage parts are linked to the user part in a wireless way. While, the MI and storage parts are linked to the provider-part in a wired way.

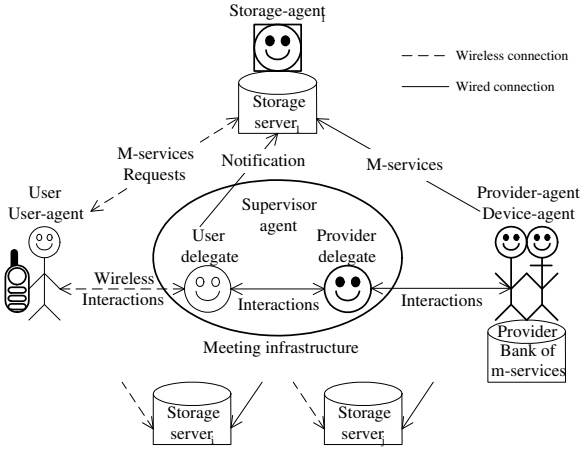


Figure1. Architecture of SAMOS

The user part consists of users and user-agents. User-agents act on behalf of users; they accept their needs, convert them into requests, and submit them to user-delegates. The MI supervisor-agent creates user-delegates. To satisfy users' requests, user-delegates interact with provider-delegates.

The provider-part consists of providers, provider-agents, and device-agents. Provider-agents act on behalf of providers of m-services; they advertise their m-services to user-delegates, through provider-delegates. In addition, provider-agents monitor the behavior of providers in case of new m-services are offered, and thus need to be announced. In Figure 1, m-services are gathered into a bank on top of which provider-agents and device-agents are located. Provider-agents create provider-delegates. In the provider-part, device-agents support the work of provider-agents. The role of device-agents is to wrap m-services before they are sent to users' wireless devices for execution. The rationale of having

device-agents is to consider the differences that exist between wireless devices, e.g. screen dimension and processor speed.

The MI part is a software platform within which user-delegates and provider-delegates reside and interact in a local and secure environment. In an open environment, most of the interactions between consumers of services and providers of services are conducted through third parties, known as brokers. Despite its important role, a broker can easily become a bottleneck. To overcome this problem, consumers and providers have to bypass the broker. In fact, they need a common environment in which they meet and interact face to face. The meeting infrastructure plays the role of this environment.

The storage part receives the sequence of m-services that will be sent for execution on wireless devices of users. According to the execution principles of SAMOS and the capabilities of wireless devices as well, m-services are sent to a user's wireless device one by one. Several advantages are obtained from the use of storage-servers. First, wireless devices have "limited" storage resources. Second, a user-agent does not have to deal with several providers to get the m-services for execution. Its unique point of contact for getting the m-services is the storage-agent. The same thing applies to device-agents that will be interacting with few storage-agents instead of interacting with several user-agents. Third, security is increased for both users and providers. Storage-servers are independent platforms where security control can be conducted in a safe way. Last but not least, statistical information can be acquired from storage-servers, e.g. what is the average time of saving services in a storage servers What is the ratio of pushing services to users' wireless devices with success/failure? etc.

4.2 Software agents

User-agent - It resides in a wireless device, acting on user's behalf. First, the user communicates with his user-agent to arrange requests. After submitting the requests to the user-delegate, the user-agent goes into a standby mode waiting for notifications from the user-delegate. These notifications concern the sequence of m-services that satisfies the user's requests. Before executing them on the device of user, m-services are put in a storage server. The MI supervisor-agent suggests to the user-delegate the storage server to be used. Several elements are taken into account in this suggestion, e.g. the storage-server's location and workload. To download the m-services one at a time from the storage server to the user's wireless device, the user-agent communicates with the storage-agent. Indeed, the user-agent keeps track of the execution of an m-service, before it asks the storage-agent to submit the next m-service. Finally, the user-agent informs the user about the completed requests.

If this is the first time that a user-agent submits requests, the supervisor-agent creates a user-delegate in the MI to be associated with that user-agent (cf. Table 1). For the next requests, the user-agent interacts directly with the user-delegate (cf. Table 2). Table 1 and Table 2 use "submit", "reply", and "notify" performatives illustrating the interactions that take place between user-agents and supervisor-agent and between user-agents and delegate-agents. In Table 1, "submit" performative has five fields among them "Device"; it specifies

the characteristics of the wireless device the user is using. The device-agent considers these characteristics in its process of wrapping m-services. In Table 2, "submit" performative occurs once a user-agent knows the delegate-agent to which it has been associated.

Table1. Interactions between user-agent and supervisor-agent

Submit(Id-submit: id_1, From: user-agent$_1$, To: supervisor-agent$_1$, Content: request$_1$, Device: (Brand: brand$_1$, Type: type$_1$, Screen dimension: dim$_1$))
Reply(Id-reply: id_1, From: supervisor-agent$_1$, To: user-agent$_1$ (In-reply-to: id_1), Delegate: (Id: user-delegate$_1$, Contact: user-delegate$_1$@...), Storage-server: (Id: storage-agent$_1$, Contact: storage-agent$_1$@...))

Table2. Interactions between user-agent and user-delegate

Submit(Id-submit: id_2, From: user-agent$_1$, To: user-delegate$_1$, Content: request$_2$, Device: null)
Notify(Id-notify: id_1, From: user-delegate$_1$, To: user-agent$_1$ (In-reply-to: id_2), Content: sequence$_1$/m-service$_{1,2}$)

User-delegate - It resides in the MI, acting on behalf of user-agent. The user-delegate receives the user's requests from the user-agent (cf. Table 2). Afterwards, it interacts with provider-delegates. The purpose is to match the user's requests with the m-services of providers. In case there is a match (it is assumed that there is always a match), the user-delegate devises the sequence of m-services to be included in satisfying the user's requests. Details on how a sequence is specified are given later on. Information about this sequence is sent to the storage-agent (cf. Table 3). The purpose is to make the storage-agent ready to receive the m-services from device-agents. Furthermore, the user-delegate notifies the user-agent about the sequence of m-services it has established for the user (cf. Table 2). In Table 3, two relevant fields of "to-get-ready" performative are important. "Destination" field corresponds to the user-agent/user for which the sequence of m-services has been devised. "M-services" field corresponds to the m-services the device-agent has to submit to the storage-agent. To set a sequence in a correct way, the storage-agent is aware of the order of m-service to be submitted by a device-agent.

Table3. Interactions between user-delegate and storage-agent

To-get-ready(Id-to-get-ready: id_4, From: user-delegate$_1$, To: storage-agent$_1$, Sequence: sequence$_1$, Destination: user-agent$_1$@..., M-services: (From: device-agent$_i$, Before: 0[m-service$_h$,device-agents$_h$]1, Id: 1[m-service$_i$]n, After: 0[m-service$_j$,device-agents$_j$]1))

Provider-agent - It resides in a provider site, running on top of the resources of the provider. A part of the software resources are m-services. Provider-delegates broadcast m-services to user-agents, using user-delegates. The provider-agent is in constant interactions with its provider-delegate (cf. Table 4). For instance: 1) it communicates to the provider-delegate the negotiation strategy it has to follow with user-delegates; 2) it informs the provider-delegate about the changes of the m-services that were announced previously; 3) it informs the provider-delegate about the new m-services to be offered; and 4) it checks the agreements the delegate-provider has made with user-delegates. At this stage of our work, agreements are not rejected. The provider-agent forwards the agreements about the use of m-services to its device-agent for actions, namely submitting m-services to a storage-agent (cf. Table 5).

Table4. Interactions between provider-agent and provider-delegate

Submit(Id-submit: id_6, From: provider-$agent_1$, To: provider-$delegate_1$, Content: $strategy_6$/m-$service_6$, Device: null)
Inform(Id-inform: id_1, From: provider-$delegate_1$, To: provider-$agent_1$, Agreement: (Sequence: $sequence_1$, With: user-$agent_1$, Service: 1[Id: m-$service_i$]n), Storage-server: (Id: storage-$agent_1$, Contact: storage-$agent_1$@...), Device: (Brand: $brand_1$, Type: $type_1$, Scrolling: yes, Screen dimension: dim_1))

Table5. Interactions between provider-agent and device-delegate

Inform(Id-inform: id_{24}, From: provider-$delegate_1$, To: device-$agent_1$ (btw of provider-$agent_1$), Agreement: (Sequence: $sequence_1$, With: user-$agent_1$, Service: 1[Id: m-$service_i$]n), Storage-server: (Id: storage-$agent_1$, Contact: storage-$agent_1$@...), Device: (Brand: $brand_1$, Type: $type_1$, Scrolling: yes, Screen dimension: dim_1))

Device-agent - Similar to a provider-agent, a device-agent resides in a provider site. The main responsibility of a device-agent is to wrap m-services according to the devices to which these services will be sent for execution. The characteristics of devices are identified during the interaction of agents and delegates, for example "Device" field of "Inform" performative (cf. Table 4). Initially, m-services are sent to storage servers (cf. Table 6). The provider-agent has already submitted the contact details of the storage-server to the device agent (cf. Table 5). We recall that the user-delegate has informed the storage-agent of that storage server about the m-services it will receive (cf. Table 3). Double-checking the information that user-delegates and provider-delegates submit to storage-agents guarantees more security to SAMOS.

Provider-delegate - It resides in the MI, acting on behalf of provider-agent. In SAMOS, the provider-delegate interacts with user-delegates regarding the m-services it offers. In addition, it interacts with its provider-agent for notification

Table6. Interactions between device-agent and storage-agent

Send(Id-send: id_{22}, From: device-agent$_1$, To: storage-agent$_1$, Sequence: sequence$_1$, Attachment: m-service$_1$, m-service$_2$)
Notify(Id-notify: id_1, From: storage-agent$_1$, To: device-agent$_1$ (In-reply-to: id_{24}), Content: ok)

purposes. Notification is then, forwarded to the device-agent for consideration. In SAMOS, the provider-delegate is obtained in two different ways: **a)** The provider-agent creates the provider-delegate and transfers it afterwards to the MI; this means that the provider-delegate is mobile; or **b)** similar to a user-delegate, the supervisor-agent of the MI creates a provider-delegate after it has received a request from a provider-agent. We privilege option **a)**. Each provider has its needs and expectations. It is more appropriate for a provider to embed the provider-delegate with the mechanisms and policies that suit him.

Supervisor-agent - It resides in the MI and has various responsibilities, among them 1) supervise the operations that occur in the MI; 2) set-up user-delegates and assigns them to user-agents, and 3) check the identity of provider-delegates when they arrive from provider sites.

Storage-agent - It resides in a storage server. The purpose of that server is to save the m-services that will be sent to wireless devices for execution. According to the information about the sequence of m-services it has received from the user-delegate (cf. Table 3, "before" and "after" fields), the storage-agent arranges the sequence as the m-services start arriving from providers. As soon as the sequence is completed, it informs the user-agent in order to make it ready (cf. Table 7). The transfer of m-services from the storage server to a wireless device is conducted in a transparent way to user. Users are not aware of the exchange happening between their devices and other remote storage-servers.

Table7. Interactions between device-agent and user-agent

To-make-ready(Id-to-make-ready: id_7, From: storage-agent$_1$, To: user-agent$_1$, Sequence: sequence$_1$-{m-service$_{1,2}$}, Status: ready)

Based on the requests it receives from the user-agent, the storage-agent pushes the m-services one at a time (cf. Table 8). These m-services are ready for execution. The deletion of m-services from the storage-servers as well as from wireless devices follows specific reliability rules (cf. Figure 4).

Table8. Interactions between user-agent and storage-agent

Request(Id-request: id_{24}, From: user-agent$_1$, To: storage-agent$_1$, Sequence: sequence$_1$, M-service: ?m-service$_1$)
Push(Id-push: id_1, From: storage-agent$_1$, To: user-agent$_1$ (In-reply-to: id_{24}), Already submitted: 3, Remaining: 2, Attachment: m-service$_1$)

4.3 Operating mode

The operating of SAMOS consists of five stages: agentification, identification, correspondence, notification, and realization (cf. Figure 2). Due to lack of space, certain stages are not explained.

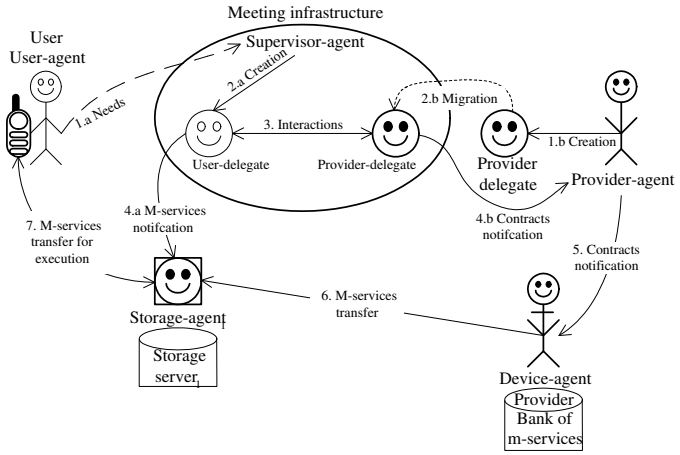


Figure2. Operating of SAMOS

Agentification stage - It aims at setting up the different infrastructures and agents that constitute SAMOS. In one hand, user-agents are established at the user level. In the other hand, provider-agents and device-agents are established at the provider level. Last but not least, the meeting infrastructure and storage servers, including storage-agents, are set up. At the agentification stage, several operations are carried out:

1. The user-agent gets aware of the user's wireless device on which it will be running.
2. The provider-agent gets aware of the resources of the provider.
3. The storage-agent gets aware of the wireless communication channels to be used.
4. The types of wireless devices to be supported in SAMOS are defined for wrapping purposes. This is undertaken for the benefit of device-agents.
5. Policies and criteria for associating users with storage servers are devised.

Identification stage - It aims at informing the MI supervisor-agent about the existence of users and providers who are interested in the use of SAMOS. At the agentification stage, user-agents and provider-agents are respectively installed on top of wireless devices of users and resources of providers. The outcome of the identification stage is the creation of user-delegates and the reception of provider-delegates arriving from provider-sites. Creation and reception operations happen in the MI. Provider-agents inform the supervisor-agent about their readiness to

submit their provider-delegates to the MI. Simultaneously, user-agents inform the supervisor-agent about the requests that users would like to submit.

Figure 3 illustrates the interaction diagram of the identification stage. What is interesting in this diagram is our representation to the actions that agents undertake. For instance, after the supervisor-agent receives the first request from a user-agent (cf. Table 1), it verifies the request and proceeds as follows: either 1) it creates a user-delegate and installs it in the MI or 2) it rejects the user-agent's request for diverse reasons, e.g. security. The same description applies to a provider-agent that aims at transferring a provider-delegate to the MI.

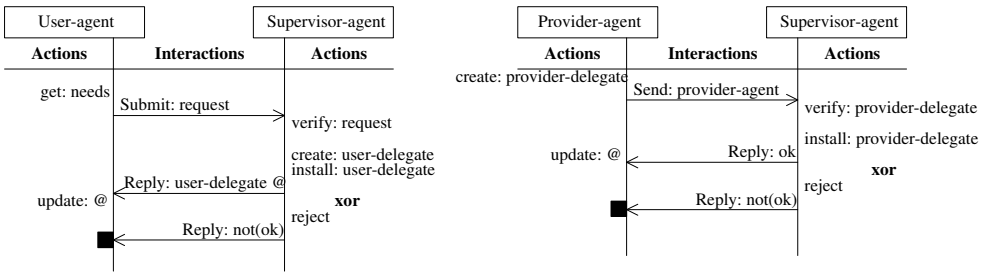


Figure3. Interaction diagram of identification stage

Correspondence stage - It allows user-delegates and provider-delegates to get together in the MI. User-delegates have requests to satisfy and provider-delegates have services to offer. First, a user-delegate searches for the provider-delegates that have the services it needs. Two approaches exist (cf. Table 9): **a)** The user-delegate asks the supervisor-agent to suggest a list of provider-delegates that have the services it needs. Here, the supervisor-agent plays the role of a recommender. **b)** The user-delegate requests from the supervisor-agent the contact details of all the provider-delegates that exist in the MI. Table 10 lists the advantages and disadvantages of both approaches.

Table9. Interactions between user-delegate and supervisor-agent

Ask-for(Id-ask-for: id_{32}, From: user-delegate$_1$, To: supervisor-agent$_1$, Content: 1[service$_i$,?provider$_i$]n)
Recommend(Id-recommend: id_1, From: supervisor-agent$_1$, To: user-delegate$_1$ (In-reply-to: id_{32}), Content: **a)** Provider-delegate$_{1,4}$ $\oplus$ **b)** Provider-delegates)

Regardless of the approach of searching for delegate-providers that is adopted, the user-delegate submits its needs of services to a selected set of provider-delegates after interactions. Based on different parameters, such as workload and commitments, provider-delegates answer the user-delegate. At this stage of our work, we assume that providers do not have services in common.

Table10. Advantages and disadvantages of searching for provider-delegates approaches

Advantages	Disadvantages
a) Provider-delegates are targeted in advance. Less interaction messages between user-delegates and provider-delegates.	**a)** Supervisor-agent becomes a bottleneck. Supervisor-agent's recommendations may not satisfy completely user-delegates. User-delegates have a narrow perception of the MI content.
b) User-delegates have a wide perception of the MI content. More freedom is given to user-delegates.	**b)** More messages to discover the offers of provider-delegates.

Consequently, there is no need for a user-delegate to look for the best service. Once the user-delegate and provider-delegates agree upon the m-services to use, notifications are sent to different recipients.

Realization stage - Its purpose is to execute the sequence of m-services that the user-delegate has devised. User-agent and storage-agent participate to the realization stage. We recall that the user-delegate has already informed the storage-agent about the different m-services it will receive from device-agents (cf. Table 3). Before the user-agent starts asking the storage-agent for the m-services it has, it waits for a notification message from the storage-agent; the notification concerns the sequence's readiness for submission.

In the realization phase, the reliability of a sequence of m-services has been taken into account. In SAMOS, a storage-server plays also the role of a back up server for the m-services that will be sent. When a storage-agent sends an m-service to a user-agent (cf. Table 8), the storage-agent keeps a copy of that service at its level. This service will be deleted when the user-agent requests the m-service that follows the one it has received. This approach works for all the m-services of a sequence except for the last m-service of this sequence; i.e. it does not have a successor. To deal with this specific case, the user-agent sends an acknowledgement message to the storage-agent so it can delete the m-service it has kept.

Figure 4 summarizes the way SAMOS deals with reliability as part of the realization stage. We consider a sequence of 3 m-services: m-service$_{1,2,3}$. Interactions between user-agents and storage-agents are illustrated with "request" and "push" performatives (cf. Table 8). In addition, actions that both agents undertake are associated with "prepare", "run", "save", and "delete" performatives. In the same figure, when the user-agent requests m-service$_2$, it is granted that the execution of m-service$_1$ has been done successfully. Therefore, the storage-agent deletes m-service$_1$. The opposite occurs with m-service$_2$; it has been requested twice. Fortunately, the storage-agent has not deleted it yet, otherwise all the sequence of m-services in the storage server could have became ineffective.

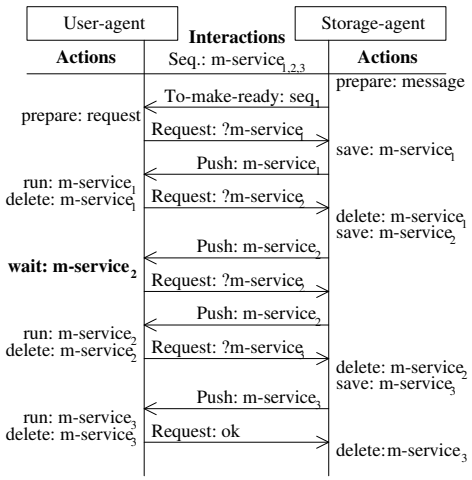


Figure4. Interaction diagram of realization phase

5 Related work

There exist several research projects that aim at studying how wireless devices can change our way of doing business and undertaking operations [1,3,7,8,11]. In HP Laboratories, [7] worked on delivering Internet services to wireless users. This work was conducted under Pervasive Services Infrastructure (Ψ) project. Ψ's vision is "any service to any client (anytime, anywhere)". The project investigated how offloading parts of applications to mid-point servers can enable and enhance service execution on a resource-constrained device. SAMOS is interested in the same issues. Furthermore, SAMOS considers how to support users in searching for m-services in an open wireless environment. The MI is part of the searching support that SAMOS suggests.

Another related work to SAMOS is presented in [8]. The Odyssey project aims at providing system support for mobile and adaptive applications. Odyssey defines a platform for adaptive mobile data access on which different applications, such as a web browser, video player, and speech recognition, can run on top. The Odyssey approach is to adjust the quality of accessed data to match available resources. In SAMOS, adaptability is considered at two different levels: 1) m-services are wrapped which means adapted to the features of wireless devices; and 2) user-agents request m-services from storage agents according to the resources that are available on the user's mobile device.

The idea of wrapping services has also been discussed in [3]. The authors suggest the use of mobile agents to adjust the content that a multimedia application transfers to clients. An agent is sent to the remote machine that provides the multimedia data (a server or an intermediary site usually called proxy) in order to adapt the transmitted data according to the constraints of the network and/or the devices of the client. In [3], the authors consider an architecture of the application that is based on client/server model. The client plays a video on a terminal; a server through a communication infrastructure provides the content

of the video. The Internet Service Provider who is responsible for routing the transmitted messages, has several intermediaries site, called proxies, on top of which additional processing can be carried out. A major element in this architecture is that the adjustment operations can be implemented on the proxies from which the video flow is transmitted without having the obligations to open the server sites to these adjustments. In SAMOS, the wrapping of services according to the features of the wireless devices of users are taken by device-agents.

6 Conclusion

In this paper, we presented SAMOS that focusses on m-services. M-services are considered as an extension to e-services. Users are more and more relying on their wireless devices to complete their daily operations. Therefore, new facilities should be provided. Our work aims at achieving "anywhere, anytime" paradigm.

Several aspects are still under investigation, among them implementation and security. For the implementation, the MI has already been undertaken in [6] using SUN's JavaSpaces technology. The development of other components of SAMOS environment is in progress. Our suggestion for security consists of enhancing the storage-agent with security mechanisms that could be generic; in the sense that such mechanisms will be applied to all m-services despite their origin (i.e. device-agent) or destination (i.e. user-agent). Specific security procedures that answer each user's priorities should be done at the wireless device level. Since the resources on wireless devices should be used in a rationale way, our future work is based on trust between user-agents and storage-agents. For instance, if a user-agent had the opportunity to deal with the same storage-agent several times and based on its previous experiences, the user-agent could entrust to this storage-agent its specific security procedures.

References

1. I. Chisalita and N. Shahmehri. Issues in image utilization with mobile e-services. In *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, MIT, Cambridge, Massachusetts, 2001.
2. M.C. Fauvet, M. Dumas, B. Benatallah, and H. Paik. Peer-to-peer traced execution of composite services. In *Proceedings of the International Workshop on Technologies for E-Services (TES'2001) held in conjunction with VLDB'2001*, Italy, 2001.
3. D. Hagimont and N. Layaida. Adaptation d'une application multimédia par un code mobile. *Technique et science informatique, Numéro spécial Agents et code mobile*, 21(6), 2002.
4. S. Helal, M. Wand, A. Jagatheesan, and R. Krithivasan. Brokering based self-organizing e-service communities. In *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems with an Emphasis on Electronic Commerce*, Dallas, 2001.
5. N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1), 1998.

6. Z. Maamar, E. Dorion, and C. Daigle. Towards virtual marketplaces for e-commerce. *Communications of the ACM*, 44(12), December 2001.
7. D. Milojicic, A. Messer, p. Bernadat, I. Greenberg, G. Fu, O. Spinczyk, D. Beuche, and w. Schroder-Preikschart. ψ - pervasive services infrastructure. Technical report, HP Technical Report HPL-2001-87, HP Laboratories Palo Alto, 2001.
8. B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, and K.R. Walker. Agile application-aware adaptation or mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, France, 1997.
9. B. Pernici and M. Mecella. Designing components for e-services. In *Proceedings of the VLDB Workshop on Technologies for E-Services (VLDB-TES)*, Cairo, Egypt, 2000.
10. G. Piccinelli, M. Sallé, and C. Zirpins. Service-oriented modeling for e-business applications components. In *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, MIT, Cambridge, Massachusetts, 2001.
11. The Ninja project. http://ninja.cs.berkeley.edu, Visited September 2001.
12. Ericom Software. http://www.ericom.com/wireless_mobile.asp, Visited Sep. 2001.
13. J. Tang and J. Veijalainen. Using agents to improve security and convenience in mobile e-commerce. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS)*, Hawaii, 2001.