Use of Agent Technology in Service and Retailer Selection in a Personal Mobility Context

Didoe Prevedourou[1], Kostas Zygourakis[1], Sofoklis Efremidis[1],

George Stamoulis[2], Dimitrios Kalopsikakis[2], Anna Kyrikoglou[2], Vassilios Siris[2],

Miltos Anagnostou[3], Lia Tzifa[3], Tenia Louta[3], Panagiotis Demestichas[3], Nikos Liossis[3],

Andreas Kind[4],

Kirsi Valtari[5], Henryka Jormakka[5], Tony Jussila[5]

[1] INTRACOM S.A.

[2] Institute of Computer Science, Foundation for Research and Technology Hellas

[3] National Technical University of Athens

[4] NEC Europe Ltd., CCRLE Berlin

[5] VTT Information Technology

Abstract

The chapter reports on work and key results of the ACTS Project MONTAGE that aims to exploit agent technology in support of personal mobility. In particular, agents are implemented to enable selection on behalf of users of the most beneficial service offer among those by multiple retailers, on the basis of user preferences, encoded in the user profile, and market offerings made by retailers. A framework for agent intelligence to support service and retailer selection is proposed and justified. Enterprise, analysis and computational models of the envisaged personal mobility context that offers the capability of agent supported service and retailer selection, are provided.

Keywords: Personal Mobility, TINA, Mobile Agents, Retailer Selection

1. Introduction

In the emerging open telecommunication market, services are traded as commodities between users and service providers/retailers. Users will be offered customized and diverse services from virtually everywhere in the world. Service competition enables them to select the most appropriate and reasonably priced service offered in the market at the price of additional complexity.

The chapter addresses personal mobility in a context of multiple competing and federated service providers. The service selection capability of a user should not be invalidated by his/her roaming in a number of service provision domains. Selection may be based on various criteria, e.g., preferred service content, desired QoS, affordable cost, etc. The chapter shows how mobile intelligent agents can be exploited in this context. A user is assumed to roam between domains without having a fixed association to a terminal. A TINA extension that supports mobility is the targeted architecture.

Mobile intelligent agents are software components having no fixed locus of operation. Typically, a mobile agent is initialized and sent off to a remote site in order to perform a specific task asynchronously. In multi-agent systems, pursuing the task involves co-operation, collaboration and negotiation with other software agents. A certain degree of autonomy and intelligence allows the agent to operate also in unpredictable or changing environments. Autonomy and intelligence can be enhanced by the agent's ability to learn from experience.

Especially in the context of personal mobility, using an active modeling paradigm can help (from a software engineering point of view) to handle the increasing functional complexity involved with service creation and deployment. For instance, access specific functionality, like finding the most appropriate service, can be encapsulated in a mobile agent and sent to where services are actually offered by service retailers. The service creation and deployment process

can therefore be focused on the service key features. The problems of distributed service offerings can be kept transparent to the service creation and selection process.

Besides the software engineering advantages of enhanced encapsulation, mobile agent technology offers further technical benefits. These technical advantages are reduced communication cost, reduced bandwidth usage, the possibility of using remote interfaces and the support for off-line computation.

This chapter is organized as follows: Section 2 presents an enterprise and an analytical model of the problem addressed. A multi-provider context in which a user can access and use services from various retailers in several domains is considered. Section 3 proposes a framework of agent intelligence to support service and retailer selection on behalf of the user. A schema for service offering and retailer selection is proposed and both the optimal and a heuristic algorithm to support the schema are described. Section 4 provides the computational view of the considered personal mobility context by defining the computational entities involved in it and their interactions during access and service sessions. Service offering and retailer selection is part of the access session. Section 5 concludes the chapter.

2. Enterprise and Analytical Models

The TINA[1] business model [6], shown in Figure 1, introduces the business roles and their inter-relationships, as they have been considered in the MONTAGE work. The ACTS project DOLMEN [1] has specialized the concept of Retailer to *Home* and *Visited* Retailer, and this specialization has been adopted by MONTAGE. For a given service, the Home Retailer is the one with whom the Consumer has a subscription contract. The Visited Retailer is the one with whom the consumer/user can establish an access session for using a service. In the following, the terms consumer and user are used interchangeably.

[1] The Telecommunications Information Networking Architecture (TINA) is an open architecture for future telecommunications and information services. TINA involves a set of principles, rules, and guidelines for constructing, deploying, and operating services.

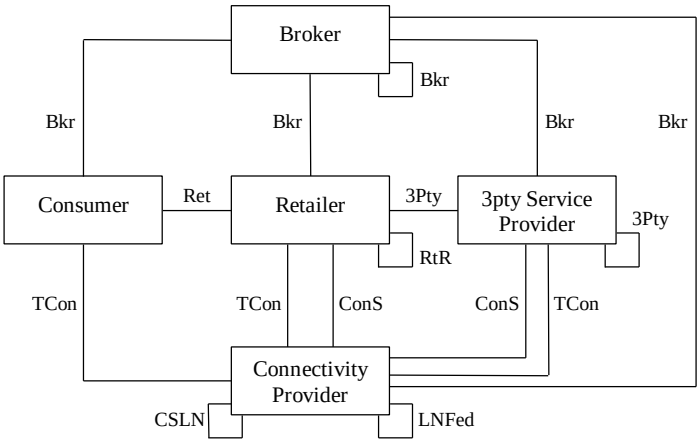

Figure 1. TINA business model.

In an antagonistic market more than one retailer may serve an area currently visited by a user. Therefore more than one retailer can offer a desired service and assume a visited retailer role. A user has certain requirements (like, QoS, cost, etc.) for a service, which are encoded in the user profile, whereas visited retailers have certain offers for the provided services. The selection of the most appropriate retailer and corresponding service offering that matches the user requirements as close as possible to the desired service is a challenging issue. The aim is to satisfy the user preferences as closely as possible, and in this respect to offer the user a virtual home environment irrespectively of his/her location. Agents can be employed for this purpose for helping the user select the most appropriate retailer.

Figure 2 shows an analysis model for the context under consideration. A user subscribes for a service type to a home retailer. The subscription is governed by a contract that contains various user preferences for the service at hand. The user makes use of a service by getting involved in a service session. The service session is controlled by a retailer, while the user preferences define the characteristics of the session. In the considered service provision scenario, a user having subscribed for a service to a home retailer, can make use of the service by contacting a visited retailer that has federation agreements with the home retailer. In this case, it is the visited retailer that actually controls the service session.

Several agents can be involved in the realization of the aforementioned scenario: A mobile user agent, called User Agent Access (UAA), carrying the user profile as part of its data, migrates from the home retailer domain to the visited domain. Since the user profile contains fields that may be dynamically changed by the user during an access session, the mobile agent completes its knowledge on the user's preferences for a particular service, by contacting the user. Then another mobile agent, called Subscribed User Agent (SUA), is created in the visited domain and gets "educated" on the user's preferences for a specific service. Then the SUA gets replicated to the nodes of all visited retailers that offer the service requested by the user and have federation agreements with the user's home retailer, and interacts and negotiates with Retailer Agents (RA) that promote the offerings of the visited retailers. Based on the results of the negotiations with the RAs, the SUA decides on the most appropriate retailer for service provision. Upon selection of the retailer, TINA-like access and service sessions are realized. The selected retailer has control of the sessions.

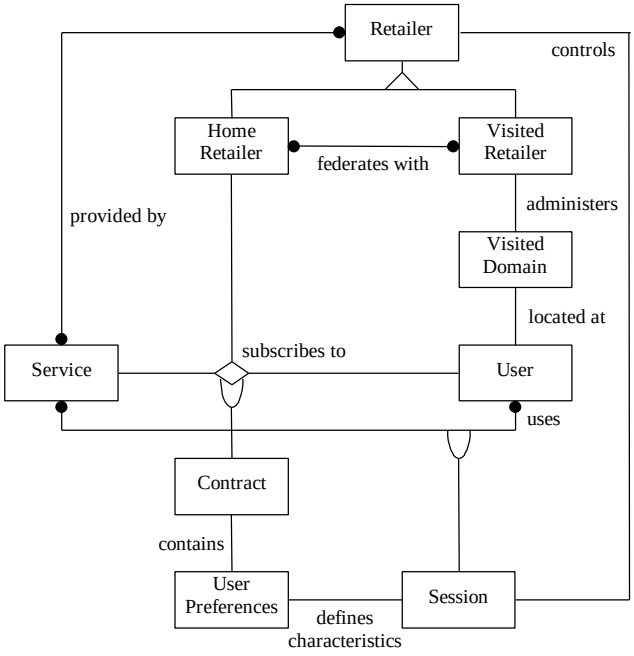

Figure 2. Information Model.

Realization of the scenario involves development of agents representing users and retailers. Such agents exhibit autonomy, intelligence, negotiations and inference capabilities. Involved

agents are made mobile in cases mobility is needed and justified on the basis of design criteria like temporal and spatial locality of communication, loose coupling and component size in accordance with [4].

3. A framework of agent intelligence for service offering and retailer selection

Agents select retailers on behalf of users on the basis of users preferences on services encoded in their user profile and of service and charge offers made by retailers. This selection is part of the access session (detailed in section 4) the result of which being that an association is established between the user and the selected retailer.

Motivation for an agent-based approach is twofold: the selection task may prove very complicated for the user, while it is more natural for the user to be represented by his own agent during the selection process rather than resort to the recommendations of the retailers. This is even more advantageous for the user if the algorithm of agent-based selection is further enhanced so that the agent continuously "learns" the parameters characterizing the preferences of the user. It should also be noted that our approach is not based on knowledge (by the User Agent) of the structure of tariffs, neither of the service combinations actually offered by the Visited Retailers; the User Agent is only based on a certain method for describing them.

3.1 User profile

The user profile specifies the services the user is subscribed to at a particular retailer, and for each service, what is the acceptable service quality, what is an acceptable/agreed cost, when the service shall be used and what limitations may be imposed to the service provision. It contains the entities shown in Table 1 (the notions of *Utility* and *Net Benefit* are introduced in section 3.2).

Information entity	Parameters	Units of expression	Entity description
Service identification	*type / category* *name* *description / keyword* *content specification*	Predefined text Predefined text User defined text User defined text	Provides the means by which the desired service can be identified.
Service media presentation	*medium type* *preference level* *encoding quality level*	Predefined text Predefined text Predefined text	Denotes the combination of media preferred and QoS levels per medium, as perceived by the user, for the selected service.
Charge framework of service	*charge range* *charge minimization* *net benefit optimization* *utility optimization*	Currency range Lowest charge Best offer Any charge	Delineates the cost the user can afford for the usage of a service.
Usage framework of service	*contiguous use* *periodic use* *scheduled use* *"ad hoc" use* *duration*	Time Time interval Time Default choice Time	Provides an outlook on service usage in respect to time, frequency and duration.
Constraints on service provision	*Retailer* *connectivity provider* *integrity* *confidentiality*	User defined text User defined text Probability Probability	Defines certain user defined criteria for service selection and limitations imposed on service provision.

Table 1. User profile.

The *service identification* is a necessity for the agent in order to know for what item it will interact. The identification may be a well-defined service name (e.g., e-mail) or a broader category in which the desired service belongs (e.g., Information Content Services) or a keyword describing the service (e.g., in category "Messaging Services" with keywords "News Group" & "Relay Chat").

The *service media presentation* provides the agent information on how the user desires the provision of the selected service.

The *cost framework* denotes how much a user is willing to pay for the provision of the service. It can be declared in terms of charge range, absolute charge, maximum or minimum charge limits. Optionally can be trusted to the agent ability to cater for the best charging offer.

The *usage framework* surmises the service usage in respect to time, frequency and duration from the user's point of view. This information can be utilized by an agent as an additional negotiation item for the best offer on service provision. The service usage can be contiguous (e.g., for one hour, 8–9 am), periodic (e.g., used every other hour from 10:30 to 14:30), scheduled (e.g., 1st and 15th of each month) or non predictable.

The *constraints* define certain user limitations to the service provision. A user may demand a particular retailer and/or connectivity provider. For the usage of the service the user can demand specific probability of surviving an attack (integrity in the use of the service) and specific probability of confidentiality. A user may also request a higher or lower importance level in the service provision, by altering the priority rank (e.g., for a service "Stock Updates" a user can request highest priority on the data delivery). The constraint list is not exhaustive and may comprise additional items.

3.2 Retailer and service offering selection schema

When selecting a retailer for a particular service provision, on behalf of a user, the agents aim to make such a selection so as to optimize the selection criterion specified in the user profile. While the "Charge Minimization" criterion is self-explanatory, the other two deserve some further explanation, which is presented below.

The *Net Benefit* for a user wishing to use a *Service* is given by:

$$Net\ Benefit(Service) = Utility(Service) - Charge(Service)$$

where the utility function $Utility(\cdot)$ has been introduced to encode user preferences for a service, such that $Utility(z) > Utility(z')$ if and only if the combination of offered service

features *z* is preferred to *z'*. It is intuitively clear that the maximum utility subject to budget constraints, i.e., under the restriction that the total expenditure does not exceed a certain amount *b*, is attained when the expenditure equals the budget constraint. However, in certain cases, it is more appropriate to assume that saving some money may also be important to the user. In such case, the objective of the user agent is to select such a combination of visited retailer and service options offered that the maximum of the Net benefit be attained; the corresponding selection criterion is "Net Benefit Maximization". If charge is of no concern at all to the user, then the "Utility Maximization Criterion" applies.

Ideally, the utility function is an ordered sequence of the possible service combinations. In other words, we should have a value of user preference for each service conjunction. In such a case, we would be able to deduce if, for example, a user prefers the service combination A = (video in low quality, audio in high quality) than the service combination B = (audio in high quality, text). This would happen if the value of the utility function of the user for the combination A, $u(A)$, would be higher than $u(B)$. The main issue about the utility function is the implied ordering according to the user' s preferences, not the specific values of utilities.

Compiling an exhaustive ordered sequence of all possible service combinations is difficult to deal with, for several practical reasons. Consider the case of a service that depends on many parameters and grades of freedom. Then, the number of possible service combinations would be very large, making the problem of computing the utility function multidimensional. Furthermore, the cost and the delay for the transport of the utility function within the migrated user agent increases, since the large number of combinations implies a big size of the utility function.

Even if the problem of complexity were solved, we would still have to deal with that of extensibility. In other words, there is no practical way to manage the utility function when more services (features) are introduced into the set of already existing ones. To be more specific, when a new parameter for a service or a totally new service is added, we would have

to find the user preferences regarding the new service combinations relatively to the already existing ordered combinations. The only way to do this in practice would be to re-compute the utilities of all possible service combinations after the enhancement, and then ask the user to give a value of his preference for each combination. This means that we should make many questions to the user in order to determine the exact order of his preferences. Furthermore, the questions to the user must be intelligent enough to deduce the correct ordering of his preferences. The point is that there is no mechanical way to find the appropriate questions. Below, we present an approximation of the utility function that is appropriate for our purposes.

3.2.1 An approximation of the utility function

One of the main difficulties in specifying a utility function associated to a vector of goods (such as the service considered) is the inter-relation among such goods regarding the induced level of satisfaction. In our case, we can assume that the contributions of independent media to the utility are additive, and we can thus adopt a simple linear-weighted model for the utility function. This model differs from the ideal model in its philosophy. Indeed, in the ideal model, we try to specify the exact ordering of the user's preferences for all service combinations, by making many arbitrary questions to the user. This is not very easy to implement correctly in practice. So, in our approach, we make a few specific questions to the user in order to determine his utility function for each service parameter. Then we use a simple way (addition) to combine the utilities for each service parameter, resulting in the utility of each service combination. Note that this way of combining the individual utilities is heuristic, yet reasonable as explained above. These assumptions are such that the benefit of the user is close enough to the ideal one, and can lead to a utility function that is appropriate for demonstrating agent-based service and retailer selection.

The parameters that are of interest in the calculation of the utility function are those that are relevant to the media presentation of the service and *not* the content selection. Individual utilities of different media are combined by computing their weighted sum. The individual utilities are produced by taking into account the preference of the user for each medium ("required", "desired", etc.) and the levels of QoS per medium (i.e., of the encoding quality levels in the user profile). They are combined through the weighted sum of the form

$$Utility(Service) = (Number\_of\_Sections) * \sum_{media} W_{medium}(level\_of\_preference) * U_{medium}(QoS)$$

where we have also included the number of content sections as a multiplicative factor.

The weight of each medium W_{medium} expresses the desire of the user for it. We define different weights for different grades of freedom. Particularly, we have:

Level of Preference	Weight
Required	3
Desired	2
Don't care	1
Excluded	–

Justification of the ordering of the weights is straightforward. Notice that "excluded" media are ignored in the calculation of the utility. However, they are taken into account prior to this calculation, in the definition of the feasible service combinations.

The utility function of QoS per medium indicates the increasing desire of the user for the increasing (better) QoS. Therefore, it is an increasing function. However, the shape of this utility function may vary. In particular, it is assumed that there are two classes of utility functions depending on whether the service is elastic (best effort) or guaranteed. In the case of the elastic services, the shape of the utility function is approximated as increasing and concave; see Figure 3.

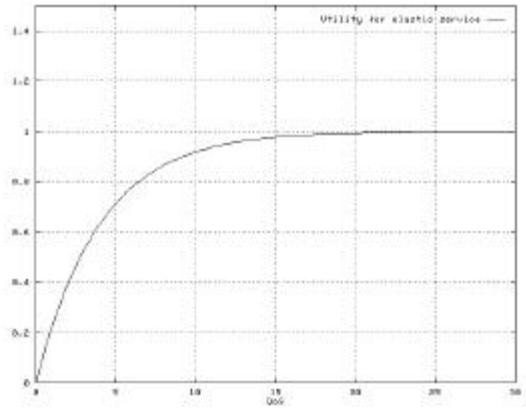

Figure 3. Utility for elastic services as a function of bandwidth.

The concavity of the utility function for elastic services (e.g., Web like services) is explained by the fact that the increment of the user satisfaction is decreasing as the QoS (bandwidth) is increasing. In other words, an additional unit of bandwidth makes more difference when the previously allocated bandwidth is small.

On the other hand, the utility function for guaranteed services is approximated as an increasing function such as that depicted in Figure 4, which is similar to a step function. The step form of the utility function for guaranteed services (e.g., video conferencing) is justified by the fact that there is a threshold below of which the user is not satisfied at all and above of which, his satisfaction is almost constant.

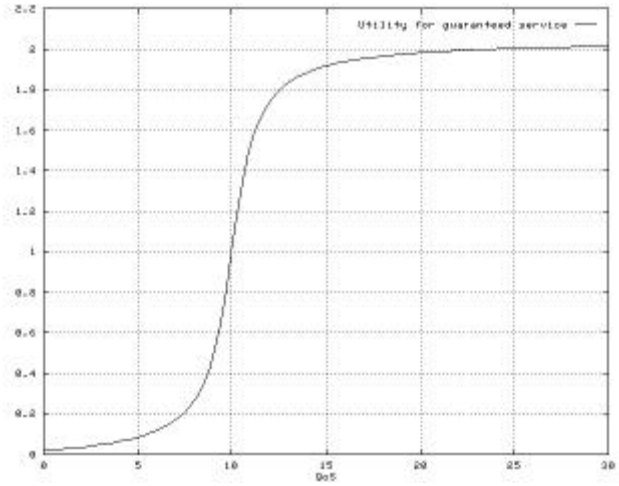

Figure 4. Utility for guaranteed services as a function of bandwidth.

Specification of Utility per medium on the basis of QoS

The utility of QoS per medium can be specified as the product:

$$order(medium) * u_{medium}(level\_of\_QoS)$$

In the above expression, the term *order*(*medium*) is the order of magnitude that is defined for each medium. This is used to translate the values of QoS utility that are given for the levels of a medium into values that are logical, relatively to the corresponding values of the other media. The orders of magnitude are meaningfully defined on the basis of studies regarding user utility [3], as well as of the relative orders of magnitude among the costs of the media. For example, the audio is about 500 times more expensive than text and video is about 7 times more expensive than audio. Specifically, taking the text as of unary order of magnitude, the following orders of magnitude are defined:

Medium	**Order of magnitude**
Video/Audio	Colored 3000 Black/White 1000
Audio	500
Still Picture	200
Text	1

The second factor of the above product, $u_{medium}(level\_of\_QoS)$, defines the utility of a QoS level of a medium relatively to the utilities of the other levels of the same medium. This definition is based on the shape of the utility function, as discussed previously. Additionally, in order to have a common base of comparing the utilities of QoS levels for different media, we make the following assumptions:

1. The utility of the minimum acceptable QoS level equals to 1 for all the media. This is a reasonable assumption to make, since the minimum acceptable level selected by the user for each medium is expected to have the same utility (per medium) for the user.

2. The utility of levels that are lower than the minimum acceptable QoS level equals to 0 for all the media. This assumption is straightforward: the non-acceptable levels have zero utility for the user.

3. The utility of levels higher than the minimum acceptable QoS level is greater than 1 and increasing. Ideally, this increase should be in accordance to the utility curve of the QoS for the particular medium and for the particular user. To avoid defining a complicated process to estimate this increase, we assume that: if the QoS goes one level up, the utility is multiplied by 1.40; for one more level up (if possible), it is multiplied by 1.25; and for one more level (if possible), by 1.15. This is motivated from the utility curves of Figure 3 and Figure 4, by the following argument: It is assumed for a user of guaranteed service that the minimum acceptable QoS level is at the saddle point (where the second derivative of the utility changes its sign); thus, regardless of the service category, the "acceptable" part of the utility function is concave, i.e., has diminishing marginal increase of utility. introducing the above multiplier, a user accepting *"low_quality"* has double the utility if provided with *"excellent_quality"*, which is justified by the asymptotic value of the curve of Figure 4.

4. In the case of text, the provision of text is the minimum acceptable requirement (one level). That is why the utility for the "level" of text medium is 1.

We now present an example to clarify the above definitions. Suppose that a user has selected a service combination of the following media and levels of QoS:

- Video/Audio with minimum acceptable QoS level of "very good quality".
- Audio with minimum acceptable QoS level of "excellent quality".
- Text (no QoS levels defined for text)

According to the above definitions, we would have:

$$u_{video/audio}("low\_quality") = u_{video/audio}("good\_quality") = 0$$
$$u_{video/audio}("very\_good\_quality") = 1$$
$$u_{video/audio}("excellent\_quality") = 1.40$$

$$u_{audio}("low\_quality") = u_{audio}("good\_quality") = u_{audio}("very\_good\_quality") = 0$$
$$u_{audio}("excellent\_quality") = 1$$

$$u_{text}(level\_of\_QoS) = 1$$

3.3 A heuristic for efficient offer selection by the SUA-replicas

In this section, we describe an algorithm that a user agent uses in order to find a good offer by the corresponding retailer. The best offer is the service combination, offered by the retailer, that optimizes the selection criterion of the user. In the case of "Utility Maximization" or "Charge Minimization", the selection of the best offer is trivial. Indeed, consider first the case of "Utility Maximization". By monotonicity, the service combination that has the maximum utility is the most demanding one. That is the one that contains all the media required by the user, even those that the user only "desires" or "doesn't care" about; moreover, the QoS level would be the highest possible for each medium. Thus, each SUA-replica has only to get offers for this service combination, and then compare them and select the least expensive one. Similarly, for the case of "Charge Minimization", by monotonicity, the cheapest service combination is the least demanding one. That is the one that contains only the media required by the user, excluding those that the user only "desires" or "doesn't care" about; moreover, the QoS level would be the minimum acceptable for each medium. Thus, each SUA-replica has only to get offers for this service combination, and then compare them and select the least expensive one. The selection is more complicated and interesting when the "Net Benefit Maximization" parameter is used. This case is treated in detail below.

The philosophy of the heuristic for net benefit maximization

An optimal algorithm for the net benefit maximization would be an exhaustive one. Specifically, the user agent would ask the retailer for the cost of all the possible service

combinations, calculate the utility of the user for each one and finally compute the net benefits. Then the user agent would return the combination that maximizes the net benefit. This exhaustive algorithm is optimal in terms of finding the optimal solution of the problem. However, the delay of finding the best solution may not be acceptable by the user. That is why we tried to find a heuristic that finds a "good" service combination (possibly a sub-optimal one) as fast as possible.

The heuristic starts its execution at a particular service combination and moves to "neighboring" combinations of the initial combination, step-by-step, as long as the computed net benefit increases. The heuristic stops if the computed net benefit is considered "good enough" for the user or does not improve.

Note that the heuristic has to maximize the net benefit subject to the following trade-off. The utility should be high enough to satisfy the requirements of the user and please him adequately, but then the charge also increases; on the other hand, as the charge decreases, so does user utility, since a less pleasing service is offered. Our approach is utility-oriented; for example, out of two service combinations that have the same net benefit, we prefer the combination with the largest utility.

The priority of the utility is accomplished by beginning the searching for the maximum net benefit from the service combination that has the *maximum utility*. By monotonicity, the combination that has the maximum utility is the most demanding one, that is the one that contains all the media required by the user, even those that the user only "desires" or "doesn't care" about; moreover, the QoS level would be the highest possible for each medium. In the sequential steps of the heuristic, the utility is decreased as long as the net benefit improves. The utility is decreased in elementary steps, that is the combination that is considered in each step is the combination that has the next lower utility than that in the previous step. We assume that if an elementary decrement of utility does not cause a significant increment of the net benefit, then it does not worth continuing negotiation. This assumption may lead to a sub-optimal

selection of the best combination that is expected to be near the optimal one in most of the cases. However, the selection is produced fast enough.

Additional speed-up of the algorithm can be attained as follows: The algorithm can decide that a certain net benefit is "good enough" so the procedure of the algorithm can stop and return this net benefit, instead of searching for a better one. If the algorithm could make such a decision, it would make it substantially faster. In order to deal with the above problem, it is supposed that the SUA keeps a history of the past optimal selections for the specific user. In this way, the SUA can compute the average of the selected net benefit, which will also be passed to the replicas. This average can be used to find a threshold (e.g., at a certain distance above the average) above of which a net benefit is accepted as "good enough".

3.3.1 Conceptual description of the heuristic

The heuristic starts by computing the combination ***S*** with the maximum utility. The SUA-replica gives this combination to the RA in order to take a charge offer. Afterwards, it computes the net benefit of this combination, as the utility of the combination minus the cost. If the computed net benefit is "good enough", then the algorithm returns the corresponding combination. Otherwise, the heuristic computes all the neighbor combinations of ***S*** in order to check if the net benefit increases when decreasing the utility.

The utility decreases when either the quality of service of a certain medium decreases or a single medium is excluded from the combination. Note that only "desired" and "don't care" media can be excluded. Based on the above observations, first-order and second-order neighbors of a certain combination can be defined. A first-order neighbor of Combination *A* is a Combination *B* in which the quality of service of a *single* medium is decreased by *one* level. Note that this neighbor is defined as long as the minimum acceptable level of QoS is not violated. On the other hand, a second-order neighbor of Combination *A* is a Combination *B* from which a *single* medium is excluded.

1. First, the first-order neighbors of ***S*** are defined. If all first-order neighbors are already considered (or if there are no such neighbors), the algorithm moves to step 2. Otherwise, the first-order neighbor with the least utility decrease (compared to the utility of ***S***) is selected, and its associated net benefit is compared to that of ***S***. If there is a substantial increment in the net benefit, the algorithm "moves" to that neighbor combination and the heuristic is repeated recursively starting therefrom. Otherwise, the algorithm continues with the first-order neighbor with the next smaller utility decrease.

2. The second-order neighbors of ***S*** are considered. If all second-order neighbors are already considered (or if there are no such neighbors), the algorithm "moves" to step 3. Otherwise, the second-order neighbor with the least utility decrease (compared to the utility of ***S***) is selected, and its associated net benefit is compared to that of ***S***. If there is a substantial increment in the net benefit, the algorithm "moves" to that neighbour combination and Step 2 is repeated recursively starting therefrom. Otherwise, the algorithm continues with the second-order neighbor with the next smaller utility decrease.

3. Return ***S***. End of the algorithm.

Note that, in the proposed heuristic it is not required to compute all the utilities of all the possible combinations of required media and QoS levels. This is because the heuristic starts by computing the utility of the "widest" combination, which "contains" the rest combinations. So the utility of a neighbor can be calculated, by simply adjusting the utility of the "dropped" parameter (QoS or medium). It should also be noted that starting from the "widest" possible combination, considering first-order neighbors first, and selecting neighbors with the least utility decrease first are all consistent with the utility-oriented feature of our approach.

Also note that if the total charge equals the sum of charges per medium, then selecting the QoS_level for each medium separately so as to maximize the net benefit per medium, and then finding the best combination of media can attain the maximum net benefit. However, this approach is restricting, in the sense that it requires that the SUA knows how the Visited

Retailers charge, while it does not leave room for non-additive charging; e.g., a Visited Retailer may wish to offer a discount to *"excellent"* quality video/audio or to a specific service combination in order to attract more users.

4. Computational View

The present section provides a computational specification of the personal mobility context considered. We assume that a user has subscribed to a set of services offered by one or more retailers. Due to administrative, historical or techno-economical reasons, a retailer offers services to users inside a domain, which can be seen as either a home or a visited domain depending on the user location. A retailer with whom the user has a subscription contract is called home retailer. The home retailer maintains for each user a profile and a set of service subscription data. Part of the subscription data is service specific, while the rest can adapt to user preferences and circumstances. Therefore, each time a user desires a service the associated parameters (e.g., QoS parameters) may be different than those of previous times, thus requiring a new search for the most preferred retailer to offer the service.

We assume that for a given service usage the user is away from his/her home domain and wishes to use a service. While in the visited domain the user registers to a specific terminal that can support the same or a similar service. The terminal is supported by a number of retailers and the user must establish an association with the most appropriate retailer for the particular service use. The communication from visited domain to the user's home domain is often expensive. A design goal has been to minimize end to end communication from the current (visited domain) location to home domain. This has led to a design, in which user agent functionality in foreign domains is carried out by visiting mobile agents.

In this personal mobility context and in accordance to the presented, in section 3.2, schema, the main points of the approach proposed for agent-based service and visited retailer selection are as follows:

1. The subscribed user agent (SUA) gets the user profile and appropriate information to enable it to compute the utility of specific service combinations
2. The SUA gets replicated to the candidate visited retailer nodes.
3. The visited retailer is assumed to have no additional information on this user's preferences stored.
4. The SUA replica at a visited retailer interacts and negotiates with the retailer agents (RA) asking one-by-one questions regarding the charge of a fully specified service combination.
5. The retailer agent offers to the user agent the best possible tariff for the service combination requested.
6. The SUA-replica assesses the net benefit and decides whether this constitutes a satisfactory choice or not; in the former case negotiation is completed, while in the latter case the SUA-replica proceeds with a new question, which is influenced by the outcome of the negotiation so far.
7. The results of the negotiations are returned to the "parent" SUA residing at the Default Retailer Domain. Thereafter a decision is made on the most appropriate retailer for the specific service usage.

Clearly in Step 4 a series of questions is posed and provides a case justifying (in accordance with [4]) agent mobility; indeed, the remote dialogue between a non-mobile SUA and the RA would take considerably more time.

The successful application of mobility with SUAs has two important requirements as regards the speed-up of the overall service selection:

1. The invocation latency of the SUA replicas incurred by unregistering/registering, serializing/de-serializing as well as transmitting the serialized agent code and state does not outweigh the actual negotiation time.

2. The data exchange between the SUA replicas and other components in the default retailer domain is much smaller than the data exchange in conjunction with negotiation between the SUA replicas and the retailer agents.

Beside the speed-up of the service selection process, it can be noted that the implementation of the SUA component is much simplified using mobility. Replication and migration turns out to be simpler to engineer than a multi-threaded (or otherwise multiplexed) client/server approach when dealing with the several retailers at the same time.

Figure 5 visualizes the overall design of personal mobility support in MONTAGE. It shows the different retailer domains present and highlights the different phases of action: access establishment, service selection, service session, and access termination. Service selection is part of the access session.

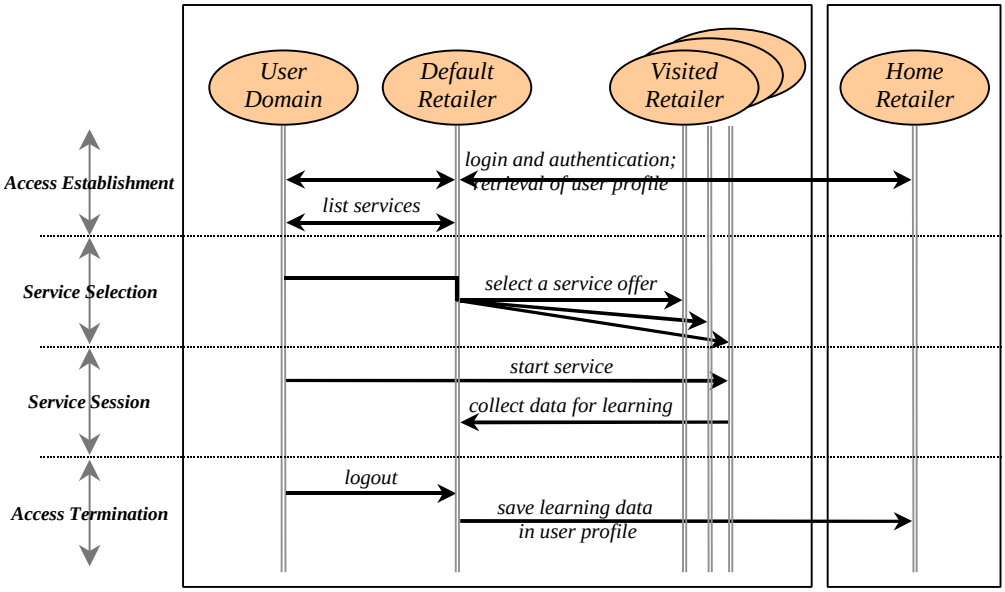

Figure 5. Access and service session scenarios integrating agent based retailer selection.

4.1 Computational components

In this section the software modules essential for supporting mobility in the presented scenario are specified. Implementation decisions and details are described as well.

To clarify the role of each module, a roadmap of software domains is presented in Figure 6. The figure gives an overall picture of the involved stakeholders and of the software modules and agents that are active in each domain. Objects sketched with a dotted line are implemented as mobile agents, the rest of the objects are stationary. The arrows in the figure mark the movement of agents between domains.

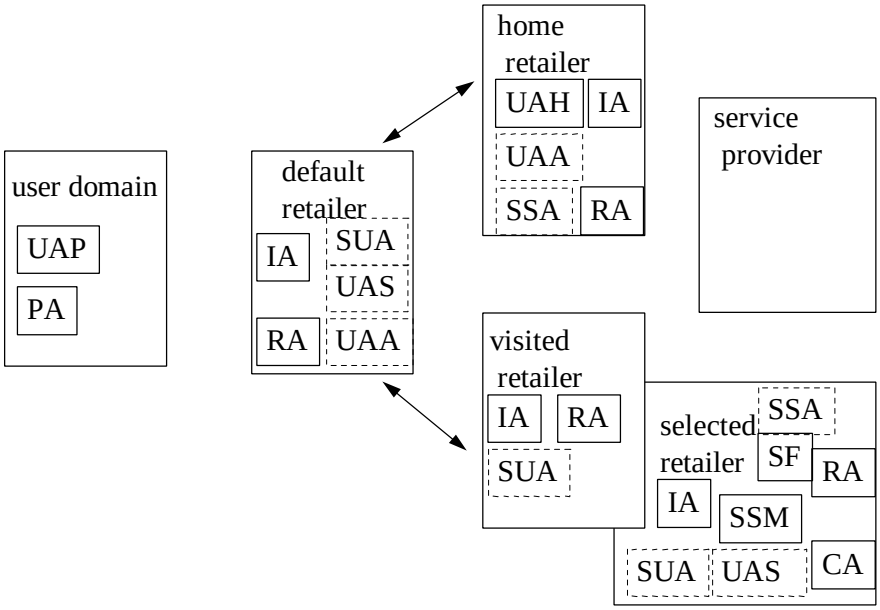

Figure 6. Software components located at MONTAGE domains.

The stationary computational objects of Figure 6 have been defined in the DOLMEN project [1] and are briefly introduced in the following, while the mobile agents defined in subsequent sections have been introduced in the context of the MONTAGE project.

- The *User Application* (*UAP*) models a variety of applications and programs in the user domain. A UAP represents one or more of these applications and programs in the TINA computational model. An *access session UAP* (*as-UAP*) allows a consumer to gain access to services.

- The *Provider Agent* (*PA*) represents the retailer in the consumer domain. The capabilities of PA support setting up a trusted relationship between the user and the retailer by interacting with the IA. Furthermore, it conveys requests for creating or joining a service session.

- The *Initial Agent* (*IA*) is the initial access point to a domain. The *IA* supports capabilities to authenticate the requesting domain and set up a trusted relationship between the domains (an access session) by interacting with the PA and also to establish an access session, allowing also the requesting domain to remain anonymous.

- The user agent home (UAH) is a stationary component residing in the home domain. It is responsible for authentication of the user and the provision of facilities for managing and updating the user profile and information of subscription in the database of the home retailer. The UAH is created when the Initial Agent (IA), in the home domain notices a login attempt and it is equipped with the user's profile and subscription information.

4.2 Mobile agents

In this section the functionality of the agents introduced in MONTAGE project is defined.

The following is a description of the data structures used in the definition of the functionality. It can be helpful as a reference when the arguments of the methods in the components definitions are analyzed.

```
module Data_Structures {

   struct AuthInfo {
      string name;
      string password;
   };

   struct ServiceItem {
      string id;
      string name;
      string ssUApClassName;
   };
   typedef sequence<ServiceItem> ServiceList;

   struct Property {
      string propertyName;
      any value;
   };
   typedef sequence<Property> PropertyList;

   /* The definitions of Property and PropertyList are
      aligned with the respective definition of the
      user's profile.
   */
};
```

4.2.1 User Agent Access (UAA)

The UAA is created by UAH when the user's home retailer notices his/her login attempt with correct information from another domain. The UAA contains the relevant parts of the user profile and also the information of the services the user has subscribed. After the creation it migrates to the domain of the default retailer to handle the user's interest and provide him the feeling of home. In a later phase the UAA is also responsible for the creation of the Subscribed User Agent (SUA), to handle the negotiations with candidate retailers for the service provision.

```
interface UAA {

   void userContext (
         in Data_Structures::PropertyList paramList);
   Data_Structures::ServiceList subscribInfoReq();
   Data_Structures::PropertyList getProfile();
   void updateProfile (
         in Data_Structures::PropertyList paramList);
   void selectRetailer (
         in long requestId,
         in string serviceName,
         in data_Structures::PropertyList paramList);
};
```

The method `userContext` is used to transmit type-independent information from the user side PA component to the UAA. This information could contain information like terminal settings needed in defining the capabilities of service provision.

The method `subscribInfoReq` is invoked by the PA and provides the user with information on subscribed services. In general a service is presented as a *struct* of three strings and when there are several of them they are presented using the IDL [5] notation of sequence. It is further assumed that this method occurs after the authentication, so no further means of providing user account or password is required. Using this method the migrated agent may be asked for the user's subscribed services and present them to the user at the terminal.

The method `getProfile` is similar to the previous one, this time the operation is on the user profile containing personal information and possibly service specific data. A profile is presented as a struct of a string telling the property name and then the CORBA type any, to

allow various representations for the actual property. Once again several of them are presented as a sequence.

The method `updateProfile` is a counterpart for the `getProfile`. It provides the user with the facilities of updating his personal information, like for instance the password used or the preferred level of service quality. The return value tells whether the update, involving access to a database, was successful or not and the properties to be updated are given as arguments. These provide also means for educating the SUA to make better service selections in the future.

When the user has selected the service he wants to use, the negotiation phase begins for finding a service offering and retailer featuring the preferred QoS parameters and associated charge. This is initiated with the method `selectRetailer`. Since it is the first half of a request-response pair, the first argument is a unique integer to identify the pair in order to avoid confusion in a multi-user environment. The second argument is a string presenting the name of the service and the last provides in a general fashion the associated parameters to be used in the search for the service.

4.2.2 Subscribed User Agent (SUA)

After the User Agent Access (UAA) has been migrated from the home domain to the default retailer domain and the user has been requested to update the service profile carried in the UAA, a Subscribed User Agent (SUA) is created. The SUA is initialized with the user profile, including the usage context (i.e., terminal equipment and connectivity) and the service profile (i.e., specification of service invocation).

The SUA creates one replica of itself for each retailer that offers the requested service in the visited retailer domain and that has a federation with the user's home retailer. The SUA replicas negotiate with Retailer Agents in the visited retailer domains in order to receive an appropriate service offer. On receipt of an acceptable offer each SUA replica informs the

initial SUA at the default domain about the offer and then terminates its execution without any further migration. The initial SUA at the default domain selects the best offer and then determines the corresponding retailer for service provision.

Since the actual negotiation between SUA replicas and RAs can be done locally without communication over a network connection, the selection process is significantly faster compared with centralized peer-to-peer communication. The selection process is furthermore increased in speed since SUA replicas negotiate in parallel.

A particular interest in the SUA implementation derived from the required integration of CORBA and mobile agent technologies. An agent platform independent solution was found that does not require explicit CORBA support by the agent platform. The solution uses a CORBA and a mobile agent server running in each visited retailer domain. After the operation `findBestVisitedRetailer()` has been invoked on the initial SUA in the default retailer domain, SUA replicas are created and sent off to the selected retailer domains using the agent migration platform. The SUA replicas arrive at the agent server and then access the retailer agent via the local CORBA name server. The RA reference is then used to invoke the negotiate operation. When an appropriate offer is returned with negotiate, an SUA replica calls `takeOffer()` on the initial SUA. The reference of the initial SUA are stored with the SUA replicas before migration to the visited retailer domains.

The SUA is specified as follows:

```
interface SUA {
   void findBestVisitedRetailer(
      in listOfEligibleVisitedRetailers retailers,
      in UserProfile userInfo,
      in string providerAgent);
   void takeOffer(
      in ServiceProfile offers,
      in RetailerInfo retailer,
      in double netBenefit,
      in long offerId);
};
```

4.2.3 User Agent Selected (UAS)

This component is used after the service selection has been completed. It is the agent representing the user in the selected domain providing it the security content of the user. It has to contain only data for the specific service, the user wants to use. Since this data is known by the UAA, it creates it with the needed data and after that the UAS may migrate to the selected domain. In the creation the UAS have to be provided with the reference to the PA component in the default domain, so that later communication is possible.

```
interface UAS {
   void serviceSessionRequest(
         in long requestId,
         in Data_Structures::PropertyList ParamList);
};
```

The interface contains only one method, `serviceSessionRequest` intended to be used when the access and negotiation phases are over. The first argument is a unique integer identifying the associated request-response pair. In the system structure it is assumed that the SUA presents a selection of the possibilities for providing the service for the user. The user may then choose one from the list and the associated QoS parameters are transmitted as a parameter in the `serviceSessionRequest` method.

4.3 Interactions between the modules

In the following, the involvement of the introduced computational objects and mobile agents is indicated in terms of Message Sequence Charts [2] modeling the envisaged access and service sessions.

4.3.1 Login sequence

The purpose of this session is to authenticate the user and grant him an access to the services to which he has a subscription at home retailer. Figure 7 describes the required operations, which proceed as follows:

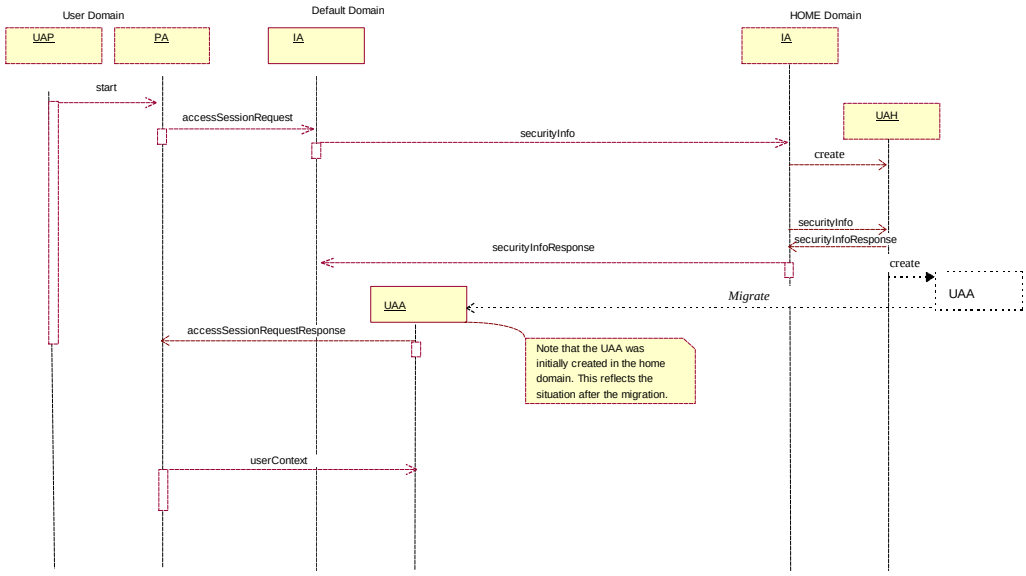

Figure 7. Login sequence.

1. The end-user requests to login into the default retailer. After filling his password, username and home retailer's name (e.g., *username@retailername*), the user presses "login" button. An appropriate User Application (UAP) at his terminal invokes `start()` operation on the Provider Agent (PA).

2. The PA sends `accessSessionRequest()` to the Initial Agent (IA) residing at the default retailer domain.

3. The IA at the default domain, after resolving the name of the IA at the home domain, sends `securityInfo()` request to it.

4. The actual authentication takes place at the home retailer. The IA (home) creates a User Agent Home (UAH) which has an access to the part of the database maintained by the home retailer containing the user's data.

5. On the IA request `securityInfo()`, the UAH authenticates the user.

6. The response `securityInfoResponse()`, is forwarded back to the default retailer.

7. In case of positive authentication, the UAH creates an User Agent Access (UAA).

8. The UAA migrates to the default domain, carrying part of the UserProfile data (S_info). S_info contains the data related to the subscribed services.

9. The UAA returns a reply (`accessSessionRequestResponse()`) to the PA. This reply includes also the UAA object reference.

10. The PA and the UAA exchange relevant information about the terminal being used (`userContext()`).

11. The UAP sends `clSubscribInfoReq()` to the PA. The request is forwarded to the UAA. As a result of this operation, the user is provided with the list of subscribed services and additionally two buttons “profile” and “setPreferences” appear to the user terminal screen.

12. When user optionally presses button “profile”, `clGetProfile()` request is send by the UAP to the PA.

13. The request is forwarded to the UAA. As a result the user obtains personal requirements on the services and service specific data.

14. By pressing “setPreferences” button, the user has an option to change his personal preferences (e.g., language being used, the time that he wants the service to be delivered etc) for the required services. Request `clUpdateProfile()` is sent to the PA.

15. The request is forwarded to the UAA as `updateProfile()`. The UAA updates its data containing the user’s preferences.

The Access Session phase is finished.

4.3.2 Service selection sequence

The purpose of the selection phase is to choose the retailer which provides the best offer for the service requested by the user. The phase starts as soon as the user has chosen the required service from the list of subscribed services and follows as described below.

1. The UAP sends `clSelectRetailer()` request to the PA. The request as `selectRetailer()` is forwarded to the UAA.
2. The UAA creates a Subscribed User Agent (SUA), an agent which will be responsible for negotiations with retailers and for the choice of a retailer offering the optimal service.
3. The SUA replicates itself and migrates in parallel to each candidate visited retailer domain where it negotiates with the Retailer Agent (RA). The list of all federated retailers has been known to the UAA.
4. SUA replicas (in the visited domains) send to the SUA that resides in the default retailer domain information of the services offered by the retailers that they visit.
5. The SUA at the default retailer domain makes the decision on the most appropriate offers and sends `selectionMadeResponse()` to the UAA.
6. The UAA sends `selectRetailerResponse()` to the PA. The selection phase ends.

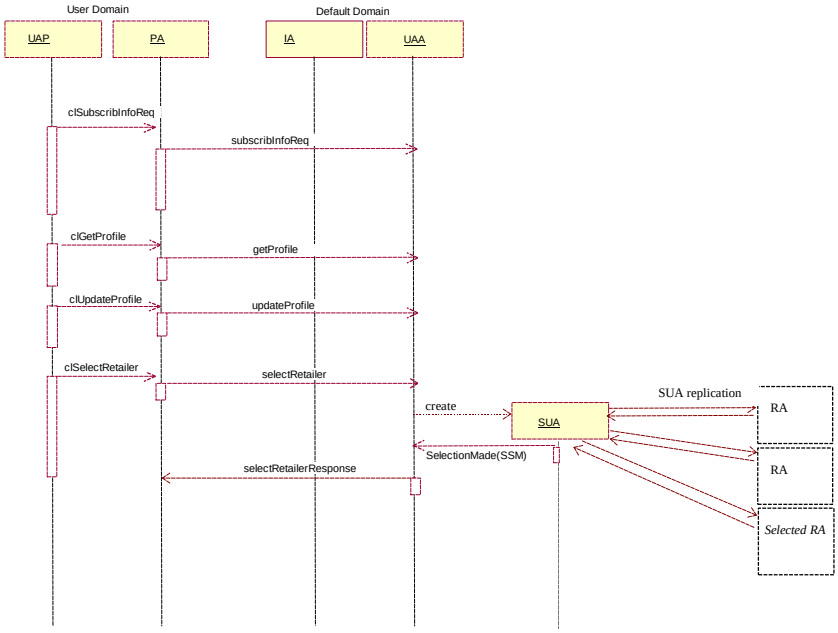

Figure 8. Service selection sequence.

4.3.3 Service session sequence

The purpose of this session is the provision of the requested service by the selected retailer to the user. An advantage offered by TINA is that the user within the scope of one Access Session can activate a number of Service Sessions.

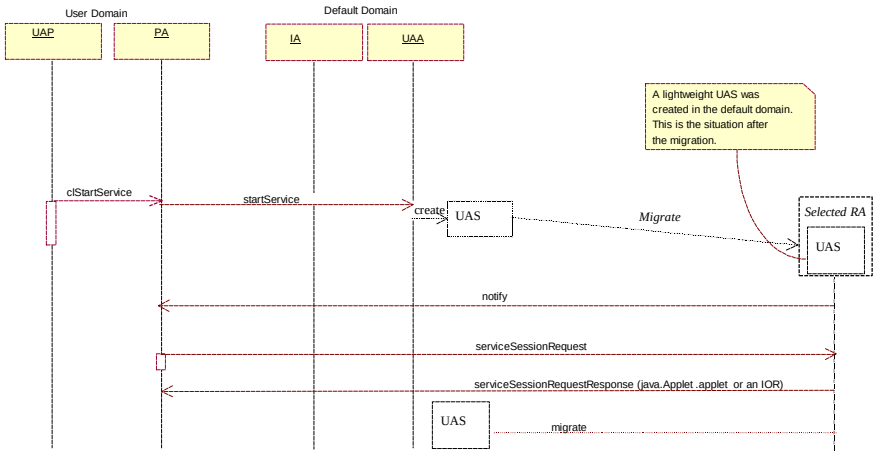

Figure 9. Service session sequence.

1. The UAP sends `clStartService()` request to the PA.

2. The request is forwarded (`startService()`) to UAA at the default domain. The object reference of the IA at the selected domain is already known to the UAA.

3. The UAA creates at the default domain a User Agent Selected (UAS) - an agent that will represent the user at the selected domain. The UAS is, in some sense, a subset of UAA —it contains information about only one service subscribed by the user, the selected service.

4. The UAS migrates to the selected domain.

5. The communication between the selected retailer and the user starts. The UAS notifies PA about its ability of providing the service.

6. The PA sends `serviceSessionRequest()` to the UAS in the selected domain.

7. The service is delivered to the user (`serviceSessioRequestResponse()`).

8. The UAS migrates back to the default domain, carrying the information about user preferences obtained during service session. The UAS forwards the data to UAA and terminates itself.

4.3.4 Logout sequence

The final phase of the interaction between the user and retailers is presented in the Logout Sequence.

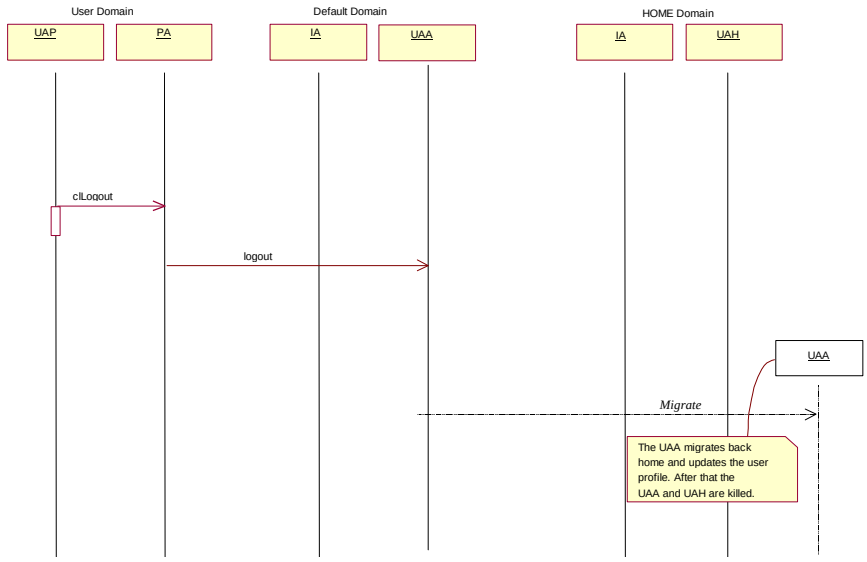

Figure 10. Logout sequence.

1. The service is completed, the user wants to logout from the terminal. The UAP sends `clLogout()` request to the PA.
2. The PA forwards this information to the UAA at the default domain as `logout()`.
3. UAA migrates back to the home domain, carrying the information about user preferences obtained since the user logged-in to the default retailer.
4. The UAA delivers the data to UAH. Then UAA terminates.
5. The UAH updates the user profile in the database at the home domain.
6. The UAH is destroyed.

5. Concluding Remarks

In this chapter, we have overviewed an approach for supporting personal mobility and beneficial service selection by means of intelligent mobile agents. The objective of the agents that are employed for the selection of the most appropriate retailer is to maximize a net benefit function, which depends on the user preferences and the cost for providing the service. The interactions that take place between users and retailers are also presented, namely, the

scenarios for logging in, service selection, service session, and logging out. Sample interfaces, as specified in IDL, of certain agents and computational objects are also presented.

A prototype implementation of the described scenario has been developed in Java with Voyager [7] as the agent platform, and application of mobility with User Agents is being validated. The most important subjects of the validation, together with the conditions that may justify mobility of the UAs as a means for making the overall service selection process more efficient involve *migration cost of the UA-replicas, temporal and spatial locality of communication and ease of implementation using mobile agents.* Validation of the intelligence framework by means of analysis and simulation experiments is presently under way. The first results are very promising, since they reveal that the proposed algorithm for service-offer selection often makes the optimal selection or a near-optimal one.

6. References

[1] AC036-DOLMEN project deliverable ASD3, *Open Service Architecture for Mobile and Fixed Network Environment*, 1998.

[2] ITU-T Z.120 Recommendation, *Message Sequence Charts*, March 1993.

[3] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*, Second Edition, High Text Publications Inc.

[4] A. Kind et al., *Towards the Seamless Integration of Mobile Agents into Service Creation Practice*, to appear in Proceeding of IS&N'99.

[5] OMG, The Common Object Request Broker: Architecture and Specification, version 2.0, July 1995.

[6] TINA-C, TINA *Business Model and Reference Points Version 4.0*, May 1997.

[7] Voyager, http://www.objectspace.com