MEMPHIS: A mobile agent-based system for enabling acquisition of multilingual content and providing flexible format internet premium services

Nikolaos Papadakis *, Antonios Litke, Dimitrios Skoutas, Theodora Varvarigou

Department of Electrical and Computer Engineering, National Technical University of Athens, 9 Heroon Polytechniou Str. Gr-15773 Athens, Greece

Received 2 August 2004; received in revised form 18 April 2005; accepted 23 May 2005
Available online 1 August 2005

Abstract

Mobile Agent technology is an interesting concept for large network-based system structures. In this paper we describe a Java-based agent environment which integrates agent execution platforms into World Wide Web servers promoting a world wide infrastructure for Mobile Agents. This system has been developed within the framework of MEMPHIS IST project, which involves the definition, development and validation of a new system for commercially significant multilingual premium services, focused on thin clients such as mobile phones, Personal Digital Assistants, etc. The concept relies on the extraction of multilingual information from various web-based content sources in different formats. The system is composed by three different Modules, namely the Acquisition, the Transformation and the Distribution Module. The content acquisition is based on dynamic user profiles. The acquired content is transformed into a meta-representation, which enables content storage independent both of source and destination format and language. The output format is based on the requirements of employed thin client terminals and the output document is distributed via the appropriate networks. The end-user receives the required information as premium services via the chosen output medium. The system supports both pull and push services. An evaluation of the system is also presented.

1. Introduction

The World Wide Web (WWW) has evolved into the most accessible and probably the wealthiest source of information of all sorts, most of it available free of charge. However, retrieving useful
information from the WWW is a complex and difficult task [1]. The explosion of the Internet has added new parameters to this problematic domain, while significantly increasing its importance in the context of e-commerce and e-business in general. Furthermore, the worldwide success of the Internet-based information services has caused an impact in the demand of more value added services to the end-users who act as information consumers. Although such services are already advanced, there is still a significant potential for further improvements. This improvement targets both the source of the information (as for instance the WWW) and the determined target—especially since the current trends indicate a migration of services to ubiquitous computing. Some problems concerning the information source are that the relevant information that the user is interested in is most times “hidden” in a labyrinth of irrelevant documents, or that the requested information is scattered over various sources, etc. On the other hand some problems concerning the information target are that the original source language may not be the user’s preferred presentation language,
or that the size and scope of information may not fit available and upcoming output devices, and others. The solution that is adopted in the MEMPHIS framework is mainly focused in (i) the collection and analysis of relevant information sources, transformations over the source information and resulting source information, in a cross-lingual retrieval scenario, (ii) the provision of an information presentation in a language chosen by the user, adjustment of the information presentation in length and complexity according to the usage situation and distribution of the information according to the required push and/or pull mode via respective networks.

The main concept is based on providing commercial services, which deliver personalized information in different languages through different media. The premium services that have been selected in this study include the acquisition, information extraction, transformation and delivery of information related to newly published books from on-line bookstores. A sample on-line bookstore is shown in Fig. 1.

To this end the provision of these commercial oriented services is achieved by combining robust state-of-the-art component technologies for automatic content acquisition, analysis, representation, transformation, and delivery using an agent-based adaptive architecture.

The high degree of flexibility for input and output formats and languages is achieved using innovative software components. These components utilize intelligent Mobile Agents (MA) which are used within Mobile Agent Platforms (MAPs) [2,3]. This way it is possible to address effective service provision and implementation of software components capable of performing intelligent tasks (e.g. searching, indexing, user notification and content delivery to diverse terminals). MEMPHIS makes use of the benefits and specific agent properties such as reactivity, pro-activity, social ability, etc.

The rest of this paper is organized as follows. The next section (Section 2) contains an overview of the system architecture and all the involved software components. Section 3 presents a detailed description of system functionality and a typical operational scenario. An evaluation of the MEMPHIS system is presented in Section 4. Finally Section 5 concludes the paper.

2. The MEMPHIS application solution

2.1. Mobile agent technology

An agent is a software computational entity that incorporates some properties, the most notable of which are (i) performing its actions with some level of pro-activity and re-activeness, (ii) acting on behalf of other entities in an autonomous fashion, (iii) exhibiting some level of the key attributes of learning, co-operation and mobility [4].

Autonomy is a feature of agents to operate without the direct intervention of humans or other entities. Another feature is reactivity meaning that agents perceive their environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, etc.) and respond in a timely fashion to changes that occur in it. Pro-activeness is another characteristic which enables agents to exhibit goal-directed behavior by taking the initiative. Co-operation is the capability agents have to interact with other agents (and possibly humans) via some kind of agent-communication language. Mobility is the ability of an agent to move around an electronic network. Learning is another important feature that concerns the capability of an agent to collect and examine huge amounts of information and to determine by itself which information is considered important to fulfill its task(s). Agents have also the capability to collect a certain amount of experience by learning over time. They can use this knowledge together with programmed knowledge in order to evaluate input and draw decisions. Stationary agents on the other hand have all the features characterizing Mobile agents except the feature of the mobility, since they are designed to consume their entire life cycle in the same Mobile Platform where they were created.

2.2. MEMPHIS architecture overview

The internal architecture of the MEMPHIS system consists of three major Modules (see Fig. 2),
namely the Acquisition Module, the Transformation Module and the Distribution Module.

1. The Acquisition Module is responsible for searching and identifying relevant documents which match a subscription profile of a given application. This is achieved by the use of MAPs that allow MAs to discover, visit, evaluate and acquire the relevant documents from various web data sources. Each Mobile and Stationary Agent is implemented as a Java class. The interaction among the Agents is performed through the standard methods provided by the MA API. Several information extraction algorithms are used by the agents of this Module [5–9].

2. The Transformation Module transforms the retrieved documents so they can match the output criteria. Dimensions of transformation include: Language Identification (determines the language of a given document), Topic Classifier (classifies documents in order to extract related topics), Summarization (different document length for each output device, e.g. the SMS service) and Translation (to provide the users’ native language; the system supports English, German and Italian). The transformation Module applies all necessary steps and makes the documents ready for delivery.

3. Finally the Distribution Module is responsible to render the content and to deliver it to the service subscribers, in due time and in due format accordingly to the specified end devices. Target devices are: PC (using fax, email both plain text and html-based) and mobiles (using SMS and WAP).

Several components are contained in the three Modules. Some of the components are designed to aim the interoperability between the Modules. Even though the different Modules work independently of each other, they share a common document database (DocDB) from which they read and into which they write. The document database contains the documents themselves and some relevant metadata like: language, topic, delivery status etc. Another shared database is the UserDB which contains information about the users that are registered into the system and their profiles and preferences as well.

2.2.1. Acquisition module

The internal architecture of the Acquisition Module is depicted in Figs. 3 and 4. This Module
encapsulates a set of Mobile and Stationary Agents. These Agents interact by exchanging appropriate information. Their main goal is to acquire information from relative web data sources.

The Agents of this Module include:

**Polling Manager**: A Stationary agent that operates in the platform on which it was created. The main task of the Polling Manager (PM) is to decide whether a new instance of a Travel Agent has to be created at a particular time. This is achieved by consulting the appropriate records in the task database. Every task consists of a time (when the Travel Agent must be created) and a set of keywords. The information to be fetched must be related to the specific keywords.

**Transaction Manager**: A Stationary agent which is responsible for instantiating and launching Travel Agents when triggered by the PM Agent. It also creates a data structure called “Suitcase”. This object will encapsulate configuration parameters and the acquired content as well. While the launched Travel Agents journey throughout the platforms where the information can be found, the Transaction Manager monitors their activity and keeps various information. For instance, it can keep information on the quantity of the Travel Agents that are waiting for launch, the number of already launched Travel Agents, their destinations, etc.

**Travel Agent**: It is the most important agent. The TA extends the MobileAgent class so as to inherit all the properties and basic functionality of a Mobile Agent (e.g., communication with the Home Platform, as well as with other Agents, etc) and also implements the TravelAgentInterface that defines some additional methods for setting the list of destinations (web pages) the TA has to visit, specifying the extraction algorithm (wrapper) to be used for fetching the information, etc. The Travel Agent can carry all the necessary parameters for conducting the acquisition process. These parameters are stored in the Suitcase object created by the
The Suitcase contains information such as keywords, an itinerary (the order that each of the web data sources has to be visited), the destination, the algorithms to be used in the search process and the location of the Home Platform in order to be able to return back, when needed. The Suitcase also holds space (appropriate data structures) for the data that will come up from the acquisition process and statistical information as well.

Data Source Broker (DSB): Another Stationary agent that maps the set of keywords into corresponding destinations where the desired information might be found. This agent takes into account some important factors in order to schedule the migrations of the Travel Agent, so that optimization of the whole system can be achieved. The DSB keeps a rated list of available data sources that are indexed: (i) by quality, in terms of extracted information usability over the amount of polled data, (ii) by content, in terms of categories and keywords that are likely to be related to the polled data and (iii) by trustworthiness. Indexing makes use of statistical information obtained, to be used further for post processing and evaluation of the content, e.g.: How popular was the extracted information? How reliable was it? Was it complete enough? What categories did it belong to?

The Search Criteria: A Stationary agent that encapsulates the knowledge for selecting the appropriate algorithms for extracting data from remote web data sources. Usually a web data source contains a variety of information and therefore according to the specified parameters different acquisition algorithms may be used.

The Ontology Agent (OA): A Stationary agent that provides ontology-related information to other agents. It contains ontologies, modeling one or more domains. It also provides a mapping between ontological concepts and keywords in various languages. It can connect to the Transformation Module to use its ontological or other semantic information, or maintain its own.

General Agent: A Stationary agent that operates in conjunction with the Travel Agents. An instance of a General Agent must be alive in every “destination” platform. When the Travel Agent arrives in a platform, it contacts the local General Agent specifying the desired web data source and the appropriate methods and algorithms to be used. At this point the General invokes one or more corresponding Service Agents to fulfill the request.

Service Agents: Stationary agents that implement search algorithms. They inherit from the Java Threads model so they can provide their services to many agents at the same time.

2.2.2. Transformation module

The Transformation Module, Fig. 5 illustrates its main components and the flow of information, follows an open system paradigm architecture which makes it easy to add modules and to replace existing modules by other ones of comparable functionality [10].

The Transformation Module interfaces with the Document Database (DocDB) for processing the documents. The User Database (UserDB) is also used to determine the target languages as required by the users. The Transformation Module consists of the following components:

The Language Identifier: which aims to determine the language of a given document [11–16]. The purpose of this is to eliminate documents in languages which are not supported by MEMPHIS.
It also triggers the language-specific resources of later components (like the classification).

The Topic Classifier: which classifies in the sequel the incoming documents according to the service topics. The topic classifier needs classification parameters [17]. These parameter files, also called models, result from a training phase. The classifier employs a naive-Bayes approach using character-level n-grams to train those models [18–20]. This training must be done for domain adaptation before runtime, using example documents tagged with one or more topics. The quality of the topic verifier runtime results increases with the size of the training corpus. The result of the training process is a model for each topic. For a given set of topics, these models can be merged into one model to improve efficiency during runtime.

The Translator: which translates text automatically from one language to another. It receives as input the text to be translated, the translation parameters such as the source and target language, the subject area of the document and then produces as output the translated text [21]. The translation system needs lexicons and grammars, for all languages and language directions involved [22]. While the grammars are fixed resources and cannot be changed, the lexicons have been adapted to the specific domains of application. The system supports the translation directions English to German, English to Italian and German to Italian in both directions.

The Summarizer: a multi-lingual system has been developed for summarization that has followed a sentence extraction approach. The summaries created by the system have an indicative abstract quality which means that they are not intended to replace the original document, but rather indicate if the original document is of interest for a reader [23–27]. The sentence extraction is based on a combination of several heuristics that are applied to the sentences of the document assigning them a relevance score. The result is the so-called summary analysis. From this analysis, summaries of different sizes can be created. Sensitivity to user provided query terms, so-called query-adaptivity, allows to generate different personalized summaries from the same text. This feature is used in MEMPHIS to provide personalized summaries by using the service and user provided subtopics from the user profiles as additional keywords to focus the summary on. Moreover after the analysis, relevant keywords can be identified. The length of the summary is variable according to user profiles and target devices (e.g. up to 160 characters for the SMS service) [28].

Notice that only if a document is in one of the languages supported by MEMPHIS and it is about one of the right topics it undergoes further processing.

The results of the different processing components are written into the MEMPHIS Document Repository (DocDB).

2.2.3. Distribution module

The internal architecture of the Distribution Module is depicted in Fig. 6.

The main components of the Distribution Module include:

The Distribution Client: which interfaces the DocDB and UserDB with the Business-to-Business server. It scans the DocDB and collects the already processed documents that are ready for distribution. Its purpose involves the creation of an appropriate XML file with all the necessary information for a successful distribution.

The Business-to-Business Server (B2B): The B2B Server listens for connections from the Distribution Client and reads the XML formatted messages. Every time a new message is received by the B2BServer, it is recorded in appropriate tables in an internal jobs database. This way the whole system can be both scalable and modular, i.e. the Distribution and the Transformation procedures may run on separate virtual machines or even different physical machines in the same or different intranets, enabling a distributed approach.

MEMPHIS Spooler (Spooler): This component scans, in regular intervals, the jobs database, that contains the messages received by the B2BServer, and processes them accordingly. For instance in the case where the message is to be sent in SMS format, a procedure implementing the communication to the desired SMS Gateway is initialized and the message is sent to the appropriate list of recipients. After a successful distribution the message is recorded in a history table.
**The Distribution Gateways**: Several Distribution Gateways have been implemented in the MEMPHIS framework. These Gateways include mechanisms for sending SMSs and emails, portals that facilitate WAP services for PDA and other thin clients.

### 2.3. The services

The architecture that has been described in the previous sections has been implemented within the framework of the MEMPHIS project. MEMPHIS provides two different ways of delivery, the push and the pull mode. In the push service the information is forwarded to the users as soon as it is acquired, or in regular intervals, based on their profiles, preferences and location. The service consists of sending out alerts or full-story messages to the subscribers. The alerts concern the availability of the respective content and the subscribers must access respective user portals (WAP-Portal for mobile phones and PDA-Portal for PDAs) to retrieve the content generated for them according to their profile. The pull service on the other hand forwards the information to the user in response to an explicit user request (real-time). This kind of service is a high-priority task: a user asks for real-time information and she/he receives it as fast as the system locates it. The target devices include: PCs (using email as a format) and mobile devices (using WAP or SMS for mobile phones and specifically rendered HTML pages for PDAs).
3. Information workflow

3.1. The user registration portal

The whole system is equipped with a web-based portal that functions as an entry point for the users and the administrators. The users' requests for information, both direct and indirect, are submitted to the system via a Web Portal and are stored in the database TaskDB. The portal is used to create and maintain the user data and their profiles. The end users interact with the system via the portal, which they can access from their PCs, PDAs or mobile phones. The portal allows the user to:

- subscribe to the system,
- subscribe to a list of available services,
- set various preferences and parameters, such as language, output devices, content, etc,
- change and update his/her profile.

It also allows the system administrators to monitor the system and obtain statistical information such as total number of users, number of requests received amount of retrieved information, etc.

Finally it allows the users to download the retrieved information or sends it directly to them via email or SMS (if they have previously subscribed to the appropriate services).

The user information is maintained in the UserDB. The Acquisition System will only search for topics for which at least one user has registered. The Transformation Module will produce only results (like translations) for which at least one user has registered. The kind of information provided to the user (e.g. information about published books) is predefined and determined by the service provider. This is because a list of web sources must be specified from which the information is to be extracted. However the service provider can dynamically allow other types of information by specifying the corresponding list of web sources to be used. Therefore, even though the presented system was applied and evaluated for the case of retrieving information about newly published books from online bookstores, it can be extended to support other applications as well.

The user interface is configurable and can be personalized to fit the user’s look-and-feel and functionality preferences. The look-and-feel is driven by separate XSLT style-sheets making easier any subsequent modification.

3.2. Typical scenario

Below follows a typical scenario in the MEMPHIS system. There must be at least two platforms in the scenario: the Home Platform (HP) and the Destination Platform (DP) that is used for the migration of the Travel Agents (see Fig. 7). There may be other intermediate platforms as well that can be used to host the Travel Agent (TA) during
its travel in the various data sources. However we will present a typical scenario that includes two platforms.

The HP may be located anywhere in the network, while the DP must be installed in the web data source’s infrastructure or as “near” as possible in order to achieve a better performance in terms of efficient bandwidth usage. By the term “near” we do not mean physical distance between the two platforms, but the ability of a fast communication between the platforms.

The following agents must be present in the HP platform: a Data Source Broker (DSB), a Transaction Manager, a Search Criteria, a Pooling Manager (PM) and an Ontology Agent.

3.2.1. The creation and preparation of the travel agent

The PM interacts with the tasks database, where all scheduled and periodic tasks are recorded. The PM is responsible for determining the various polling frequencies, and “collecting” all the relevant records. Polling can be executed either in respect to a specific web data source, or for a specific context (i.e. a specific category or fact). Polling frequencies are determined from statistics about: the kind and volatility of information (category etc.), its importance, its popularity (number of hits, number of users explicitly asking for it), the amount charged for it etc. The PM notifies the Transaction Manager (TM) to create a new instance of a Travel Agent, providing also the keywords or categories to search. This data is placed in the Suitcase object, that TM creates for every Travel Agent. The TM also has to perform some additional tasks. It first interrogates the DSB which keeps a rated list of available web sources that are indexed by the factors that have been mentioned in the DSB description. It provides the TM with a list of alternative web data sources and eventually an itinerary based on these web data sources. There are several constraints and factors that must be taken into account such as the required quality of information, the type, usability, statistics of data from each data source and the potential cost of data extraction for each source. The DSB must tackle the combinatorial problem of optimizing all these factors.

The DSB is also supported by a domain ontology that maps concepts or categories to keywords and models relations between concepts. The additional benefit by this fact is that it allows the construction of an enriched set of keywords about concepts related to the search.

At this point the TM has the knowledge on where to send the Travel Agent and what it has to search for.

The next step is to determine which search methods the Travel Agent has to use. This is achieved by the Search Criteria Agent which provides a set of appropriate acquisition algorithms suitable for the specific destination(s).

The TM now holds all the information needed to launch the Travel Agent. It also registers it into a “pending list” by assigning a unique Identification Number to it. The Transaction Manager is responsible to launch the Travel Agent out of the HP, to receive it when it comes back and to update the pending list accordingly.

3.2.2. The travel agent migration

When the Travel Agent arrives at the Destination Platform (DP), it requests a specific Stationary agent, the “General Agent”. The General Agent must be present in every DP. Its existence is imposed by security issues, since this way every Travel Agent is given a limited execution environment (sandbox) [29,30]. The boundaries of this notional box limit the scope of a malicious program to cause damage to the computer system. Depending on the particular search methods to be used, an existing Service Agent is triggered by the General Agent. Recall that each Service Agent implements a set of information extraction algorithms (wrappers) [31,32] that are capable of extracting data from the specific web data source. In the sequel the acquisition process starts. When these processes are completed, the Stationary Service Agent adds the resulting information into the Suitcase of the Travel Agent. When all the Travel Agent’s requests have been served in a specific DP, it eventually migrates to the next DP and so on. If no further destinations can be found, it returns to its home location at the HP. After its arrival into the HP it transmits the acquired contents of the Suitcase to the Transaction Manager which forwards them to
the next Module, namely the Transformation Module. The Transaction Manager is informed of the Travel Agent’s arrival and starts the removal procedure of the corresponding Travel Agent ID number from the list of launched agents. Thus the Travel Agent’s life cycle is concluded.

3.2.3. The Transformation of the acquired content

The Transformation takes place on the acquired information by reading the new documents from the DocDB, calling the processing functions according to the user profiles, and writes the results in the form of annotated documents into the DocDB. In particular, each incoming document is first processed by the Language Identification component, which identifies the document language. If the language is not one of the languages supported by the system, the document is discarded. Else it enters the Topic Classification component, which assigns it to one or more relevant categories. This is necessary in order to distribute each document only to users that may be interested in it. The next step is to translate the document into other languages so that each subscribed user that may be interested in it will receive it in the language of his/her preference. Finally the document is processed by the Summarization component. This is required because in many cases it is not possible to send the full document due to restriction imposed by the user’s device or the preferences of the user himself. In fact the Summarization component does not produce a fixed summary of the document, because the final document length may vary. To provide flexibility, it combines several heuristics to assign a relevance score (weight) to each sentence of the document. In this way, when a document is sent to a user, a summary can be dynamically created according to the length limitations of the particular case. This summary is created by inserting sentences in order of relevance score checking at each iteration that the specified limit is not exceeded.

3.2.4. Distributing the Information to the users

The Distribution Client component is the intermediate of the Transformation and the Distribution Modules. The Distribution Client component implements a set of algorithms that match documents found in the document database with users from the UserDB (see Fig. 6) [33]. These Matching and Prioritization Algorithms use both deterministic and probabilistic techniques to achieve the selection of users concerned in particular documents. These algorithms apply both in push and pull services. The Distribution Client component scans periodically the DocDB for newly acquired documents. If a new document is found, then it determines the users that are interested in the given document. This is achieved by using a suite of both probabilistic and deterministic algorithms. Deterministic algorithms use information about the user, which has been explicitly entered by the latter i.e. interesting categories of documents, keywords, topics etc. The probabilistic algorithms bring into play implicit or mined information about a particular user and his/her preferences. This is achieved by performing regular user classification and using data mining techniques. This way the system can determine a list of users that are interested in that specific keywords or categories mapped with the particular document. The Distribution client also includes a simple filter, which filters messages that exceed the total number of messages the user prefers to receive per day. The filter can even create digest messages from two or more messages if it finds they are related. After that a set of lists with users is created. Every list is populated with users that are interested in the particular document and have explicitly asked for a specific language supported by the MEM-PHIS system. The document exists in the Document database in various versions, each one translated in one of the supported languages in XML format. Table 1 shows a typical XML file output from the Transformation Module. Every user in each one of these lists must be processed at this point. If a user in his personalization settings has asked to receive the full version and his device is capable of accepting it, then all the sentences contained in the XML file will be used for the creation of the message to be sent. In case the user has asked a short version of the document or the output device does not support full versions (e.g. a user that receives documents in SMSs through his mobile phone), then only some sentences included in the XML file (see Table 1) must
be selected for the output. The system supports variable length for the summarized messages in order to comply with the needs of each individual user or device. The selection of the sentences is based on the weights defined by the Summarization component during the Transformation phase. For instance, in the case of a user that receives SMSs in his mobile phone, the length of the output is restricted to 160 characters. This can be formulated as a typical Knapsack problem [34,35]. Our implementation solves this problem by implementing a backtracking approach to the 0/1 Knapsack problem. The Distribution Client also creates a new XML containing the text to be sent and the list of recipients (mobile phone numbers or email addresses). Table 2 shows a typical XML file created by the Distribution Client with two mobile phone numbers in the recipient tag. At this point the Distribution Client has to communicate with the B2BServer in order to transmit the created messages. Notice that there are a lot of created XML files, each one containing full or summarized versions, in different languages and intended to different devices according to the specific user groups interested in the specific document. The B2BServer acts as a server accepting a socket con-

Table 1
A typical document in XML format after the summarization procedure

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<BOOK doc-lang = "en">
  <TITLE>The Rise and Fall of Rock Radio</TITLE>
  <SUBTITLE></SUBTITLE>
  <AUTHOR>Richard Neer Foreword by Steven Van Zandt</AUTHOR>
  <PUBLISHER>Villard</PUBLISHER>
  <RESELLER>Random House</RESELLER>
  <YEAR>2001</YEAR>
  <PRICE_ORIGINAL>$19.00</PRICE_ORIGINAL>
  <LANGUAGE>en</LANGUAGE>
  <SUMMARY_ANALYSIS>
    <tu id = "1" weight = "6.77"">It was all so honest, before the end of our collective innocence.</tu>
    <tu id = "2" weight = "7.0"">But on FM radio it was all spun out for only you.</tu>
    <tu id = "3" weight = "5.51"">As a young man, Richard Neer dreamed of landing a job at WNEW in New York—one of the revolutionary FM stations across the country that were changing the face of radio by rejecting strict formatting and letting disc jockeys play whatever they wanted.</tu>
    <tu id = "4" weight = "5.87"">FM: The Rise and Fall of Rock Radio chronicles the birth, growth, and death of free-form rock-and-roll radio through the stories of the movement's flagship stations.</tu>
    <tu id = "5" weight = "5.53"">Full of fascinating firsthand stories, FM documents the commodification of an iconoclastic phenomenon, revealing how counterculture was coopted and consumed by the mainstream.</tu>
  </SUMMARY_ANALYSIS>
  <SOURCE>/home/nkpap/memphis/english/randomhouse.com/rh1.xml</SOURCE>
</BOOK>
```

Notice the weights (in the “tu” tags) assigned to each sentence.

Table 2
A summarized message to be sent as an SMS

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<XML>
  <REQUEST SENDMESSAGES />
    <CUSTOMER>USERNAME=nkpap/USERNAME</CUSTOMER>
    <PASSWORD=nkpap/PASSWORD>
      <CUSTOMER></CUSTOMER>
      <SUBJECT>MEMPHIS Alert</SUBJECT>
      <RECIPIENTS>
        <PHONE>306976675605</PHONE>
        <PHONE>306975577685</PHONE>
      </RECIPIENTS>
      <MESSAGE></MESSAGE>
    </REQUEST>
</XML>
```

A connection from the Distribution Client in order to receive the aforementioned XML file. After parsing the received XML file, the B2BServer stores the extracted information into a relational database. This is the database that the Memphis Spooler component scans periodically. The Spooler implements a queue, called Output Document Queue, and also contains a subcomponent called Distribution Selector. Whenever a record is found, the Distribution Selector selects the appropriate Distribution Gateway taking into account parameters like service quality, cost etc. The Gateways supported by the system provide SMS, Email, Fax and WAP services. For instance, if a message must be sent to a user through an SMS Gateway, the cheapest SMS Gateway for the country of the particular user is used (see Fig. 8).

As follows from the above description, the MEMPHIS implementation has followed a modular approach. The Distribution Module may be running on the same machine as the Transformation Module or on a different machine and/or network.

4. System evaluation

4.1. Software issues

The Mobile Agent platform used in the MEMPHIS system is Grasshopper [36]. The Grasshopper core system is an extensible pure Java-based Mobile Agent platform, supporting both Stationary and Mobile Agents. It provides all necessary functional capabilities to develop and run agent applications. Grasshopper is compliant with the mobile agent standard of the Object Management Group (OMG) [37], i.e. the Mobile Agent System Interoperability Facility (MASIF). The MASIF standard has been initiated in order to achieve interoperability between mobile agent platforms of different manufacturers. Grasshopper is also compliant with the Foundation of Intelligent Physical Agents (FIPA) [38] specification through a software package, referred to as “Open FIPA Add-on”. The core system package enables customers to set up a Grasshopper-based system environment which consists of several Grasshopper agent systems (agencies), grouped within a domain (region), such as an intranet.

4.2. Runtime results

Mobile agents enable the usage of an architecture which comes sometimes in contrast to that of traditional acquisition and search engines. In the traditional acquisition engines, the data that has to be searched are transferred from their source site to the search engine’s home site, where they are then processed. Data found to match the search criteria are presented to the user or kept for further processing, whereas those not matching the search criteria are discarded. Typically, data that match the search criteria will only be a small fraction of those found in the source site.

With MA, instead of transferring data to the search engine for processing, we can transfer the search engine itself to the data source and have it execute the search algorithms locally on the data source. The advantage is that there is a potential for substantial bandwidth saving. In the traditional, centralized search model, all the source’s data have to be transferred from the source to the search engine’s site. With the proposed distributed search model, the search engine (or at least part of it) has to be transferred to the data source, but only the matching data are then transferred back to the search engine’s site. In all realistic situations, the search engine’s size in bytes is negligible compared to the size of the source’s full data.

Fig. 8. WAP navigation within a thin client.
collection, so that the saving in bandwidth is counted in orders of magnitude. Let us denote by \( S \), \( S_R \) and \( S_T \) the size (e.g. in KBs) of the total amount of information contained in the web source, the amount of information contained in the web source that is related to the given query and the size of the TA respectively. In all typical cases \( S_R \ll S \) and \( S_T \ll S \). In the case of a centralized architecture, all the information must be transferred to the local system in order to extract the relevant content. Therefore the size of information transferred over the network is \( S \). However, using a MA architecture, the TA is sent to the web source, the extraction takes place there and the TA returns to the home platform carrying only the relevant information, thus the traffic in this case is \( S_R + 2 \times S_T \), which is significantly lower than \( S \). This approach allows for a better exploitation of the Mobile Agents capabilities that is suitable for the business cases addressed by the MEMPHIS project (see Fig. 9).

A very simple but indicative example is illustrated in Table 3. The Web source that has been selected for the example is the German online bookstore www.buecher.de. This site has been regularly updated with new books that were acquired based on the preferences set by the users. By following the above differential approach, only the new books that are of interest are transmitted from the DP to the HP. At the beginning of the test the total number of books-concerning a particular category was 237. It can easily be seen how the

<table>
<thead>
<tr>
<th>New books on site</th>
<th>Amount of downloaded information (in bytes)</th>
<th>Percentage of the total load (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>237</td>
<td>1,627,803</td>
<td>100.00</td>
</tr>
<tr>
<td>213</td>
<td>1,458,378</td>
<td>89.59</td>
</tr>
<tr>
<td>190</td>
<td>1,301,213</td>
<td>79.94</td>
</tr>
<tr>
<td>166</td>
<td>1,146,169</td>
<td>70.41</td>
</tr>
<tr>
<td>142</td>
<td>984,399</td>
<td>60.47</td>
</tr>
<tr>
<td>119</td>
<td>824,472</td>
<td>50.65</td>
</tr>
<tr>
<td>95</td>
<td>660,055</td>
<td>40.55</td>
</tr>
<tr>
<td>71</td>
<td>496,145</td>
<td>30.48</td>
</tr>
<tr>
<td>47</td>
<td>329,718</td>
<td>20.26</td>
</tr>
<tr>
<td>24</td>
<td>162,270</td>
<td>9.97</td>
</tr>
</tbody>
</table>

Fig. 9. A decentralized Agent-based Acquisition system.
amount of downloaded information significantly reduces as the portion of the new books reduces.

During the validation phase of the Acquisition Module in the MEMPHIS project, the agent-based environment described above has been setup and used for several months. Over 3 Gigabytes of documents have been acquired from 30 different web data sources, which formed totally more than 65,000 HTML pages. Concerning the overall system, two demo services, called “MediAlert” and “FinAlert” had been implemented. These services offered alerts on newly published books and financial information regarding bank merging, respectively, and were both available via web, WAP-enabled mobile phones, Personal Digital Assistants (PDAs) and PCs. The testing was conducted at different countries and included focus group sessions on the usability aspects of the demo services. The test group consisted of a total of 30 participants. The participants evaluated the system services as quite satisfactory in terms of the frequency of delivered messages and the quality of the received content.

5. Conclusions

In this paper we have presented a Java-based agent environment that has been designed within the framework of the MEMPHIS project, which aims to provide prime services to end-users using both rich and thin clients. The service refers to an automated acquisition of new published books in specific sites based on criteria that have been set by the users at their registration phase. The system has followed a modular architecture comprising from three modules. The Acquisition Module acquires the documents from the various web data sources. The Transformation Module processes the acquired information and transforms it according to the user requirements. In particular, this process includes identifying the language and the topic of each document, translating them to other supported languages and creating summarized versions of the documents. Finally, the Distribution Module provides the information to the end-users either in a push or pull mode through the appropriate gateways. The system has been designed, developed and validated in the framework of the MEMPHIS IST project. During the evaluation phase of the system, the services provided to real end-users have been assessed in terms of usability, efficiency and accuracy of the information that belong within the given interests of the users. The adoption of the Mobile Agent approach has performed in an efficient manner enabling the utilization of the described architecture for commercial oriented services. It is important to point out that the benefit of deploying an agent approach as an architectural solution for such complex systems relies on the fact that it is efficient in terms of bandwidth utilization. This substantial bandwidth saving derives from the fact that instead of transferring all the data from the source to the acquisition engine for processing, as in the traditional, centralized search model, we can transfer the acquisition engine itself, as a Mobile Agent, to the source, process the information locally and return to the home platform carrying only the data matching the required criteria.

Acknowledgments

The work that has been performed was partially funded by the MEMPHIS project (IST-2000-25045). The authors would like to thank the MEMPHIS [39] consortium for their fruitful cooperation.

References

[38] Foundation of Intelligent Physical Agents (FIPA), <http://www.fipa.org/>.
Nikolaos Papadakis received the Diploma degree in Electrical and Computer Engineering in 2000 and the Ph.D. degree in 2005 from the National Technical University of Athens (NTUA). Since 2000 he works as research associate in The Telecom Lab of the National Technical University of Athens. He has been involved in several national and EU research projects. His research interests are focused on data mining, semantic web, Information extraction and machine intelligence.

Antonios Litke received the Diploma from the Computer Engineering and Informatics Department, University of Patras, Greece in 1999. Currently, he is pursuing his Ph.D. in the Telecommunication Laboratory of Electrical and Computer Engineering Department of National Technical University of Athens and works as research associate in the Institute of Communication and Computer Systems participating in numerous EU and National funded projects. His research interests include Grid computing, resource management in heterogeneous systems, Web services and information engineering.

Dimitrios Skoutas has received his Diploma from the Dept. of Electrical and Computer Engineering of the National Technical University of Athens in 2003. He is currently a Ph.D. candidate at the same department, where he also works as software engineer and research associate. His research interests involve semantic web technologies, information retrieval and extraction from web sources, data mining, web services and grid computing.

Theodora Varvarigou received the B. Tech degree from the National Technical University of Athens, Athens, Greece in 1988, the MS degrees in Electrical Engineering (1989) and in Computer Science (1991) from Stanford University, Stanford, California in 1989 and the Ph.D. degree from Stanford University as well in 1991. She worked at AT&T Bell Labs, Holmdel, New Jersey between 1991 and 1995. Between 1995 and 1997 she worked as an Assistant Professor at the Technical University of Crete, Chania, Greece. Since 1997 she is working as an Associate Professor at the National Technical University of Athens. Her research interests include Grid Technologies parallel algorithms and architectures, fault-tolerant computation, optimisation algorithms and content management.