Research Repository UCD

Title	Service fusion in mobile contexts
Authors(s)	Shen, J. M., O'Grady, Michael J., O'Hare, G. M. P. (Greg M. P.)
Publication date	2010-12
Publication information	Shen, J. M., Michael J. O'Grady, and G. M. P. (Greg M. P.) O'Hare. "Service Fusion in Mobile Contexts." Springer, 2010.
Publisher	Springer
Item record/more information	http://hdl.handle.net/10197/2795
Publisher's statement	The final publication is available at springerlink.com
Publisher's version (DOI)	10.1007/978-3-642-13526-2_3

Downloaded 2023-04-07T10:52:29Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

Service Fusion in Mobile Contexts

J.M. Shen¹, M.J. O'Grady², G.M.P. O'Hare²

¹School of Computer Science & Informatics, University College Dublin (UCD), Belfield, Dublin 4, Ireland.

² CLARITY: Centre for Sensor Web Technologies, University College Dublin (UCD), Belfield, Dublin 4, Ireland.

sjm2218@hotmail.com, {Michael.J.OGrady, Gregory.OHare}@ucd.ie

Abstract. Service provision for mobile users poses several challenges for those managing enterprise information systems. Some of these are ubiquitous, for example, privacy and security. Others give rise to integration and interoperability issues. The agent paradigm, though offering potential in fixed networked scenarios, has not been seriously considered as a mechanism for incorporating mobile clients, primarily to their computational intractability on mobile devices. However, recent developments in agent and mobile computing technologies – both from a hardware and software perspective, have rendered the mobile phone a viable platform for agents. One implication of this is that agents now offer a viable platform for service delivery to mobile users. This chapter reflects on some of the pertinent issues necessary for this, including Agent Oriented Software Engineering (AOSE). The discussion is illuminated with a systematic description of EasyLife – a framework that harnesses the agent paradigm to construct and deliver services to mobile users.

Keywords: Ambient Intelligence, Mobile Computing, Agent-oriented Information Systems, Location-aware computing.

1 Introduction

As the boundaries between mobile telecommunications and computing continue to blur, the demand for access to services and data from mobile devices increases. This development is not surprising; the process has been ongoing since the production of the first laptop, or at least one augmented with a wireless modem. However, it can give rise to significant problems to service providers of various hues. As an example of this consider the difficulties experienced and the energy expended in making the internet accessible to mobile users. The difficulties are exacerbated by the nature of the archetypical mobile device. When compared with the traditional workstation or desktop PC, a mobile device is not as computationally powerful, possibly several orders of magnitude smaller. Nevertheless, many people prefer to use their mobile phone for service access. Though the experience may be leave something to be desired, the convenience of anytime, anywhere access compensates in many instances.

Enterprises are not immune to developments in mobile computing, and the option of being able to access company data remotely is one that executives and sales

professionals amongst others find particularly useful. Of more interest to enterprises however is revenue generation, and providing services to the mobile computing community. Such services may be specialised in nature, or may demand cooperation with other enterprises or third-part service providers. Either way, it frequently demands a radical review of business and IT management practice.

2 The Quest for the Enterprise Mobility

Enabling Enterprise mobility and mobile service provision demands an innate understanding of the mobile computing domain. A firm grasp of its limitations must also be obtained if these are to be circumvented.

2.1 Mobile Computing Paradigms

Two approaches exist for supporting mobile users. In the first instance, an infrastructure may be embedded in a physical environment such that computational resources can be accessed as needed - ubiquitous computing adopts this approach. In the second instance, the user carries the necessary computational apparatus about their very person. Wearable computing would be an example of such an approach. However, for many people, conventional mobile telephony would be the de facto interpretation of mobile computing.

A more recent interpretation of mobile computing is Ambient Intelligence (AmI) [1]. AmI incorporates ubiquitous computing and seeks to provide seamless and intuitive interaction to services. Though an environment may be saturated with embedded computational artefacts, enabling intuitive interaction with such artefacts is essential if the services provided by the environment are to be availed of. Intelligent User Interfaces (IUIs) [2] are proposed as a means of achieving such interaction. How the intelligence required for such interfaces may be realised in practice for embedded devices, or indeed lightweight mobile devices such as mobile phones is an open question. One computationally tractable solution is that of embedded agents [3]. Though such agents may drive the logic for the IUI, there is no reason why they cannot be harnessed for other purposes, and form the mobile component of enterprise-based Multi-Agent Systems (MASs). When viewed in this light, it can be seen that such agents offer an apt solution in Enterprise computing, enabling the construction of innovative services in Business (B2B), Business to Consumer (B2C) and Business to Employee (B2E) scenarios.

2.2 The Mobile Phone – Services on Demand

The mobile phone represents the default mobile computing platform of choice for many people. From an enterprise perspective, a case could be made for the laptop; however, in comparison with the fixed workstation, the wireless connection with its poor latency and data rates is the key differentiator. In contrast, the situation is more complex with mobile phones. The limited screen real estate, typically $\frac{1}{4}$ VGA, as well

as the unwieldy methods for interaction, usually a combination of 5-way navigation pad, soft keyboard and a compact alpha-numeric keypad, are characteristic of mobile devices, and are not designed for long interaction sessions. These characteristics, or from a software engineering perspective – constraints, significantly influence the design of services and may be the sole cause of re-engineering the user interface component. Another constraint is the paucity of raw computational resources available on such devices. Again, this has many implications, one being that the ability to manipulate and visualise data on the device is seriously compromised. Thus service composition and adaptation, for example, personalisation, usually occurs on fixed network servers.

2.3 The Question of Heterogeneity

Mobile devices are manifested in a multitude of capabilities and form factors. This heterogeneity may be perceived as a great strength as the needs of many market segments are addressed; the commercial success of mobile telephony is testimony to this. From a service provider's viewpoint, and indeed that of software engineer, heterogeneity introduces major complexity into the design and implementation process. Two approaches can be considered. In the first case, a service can be designed for the lowest common platform specification. In this case, the customer base is widened, but the potential for service differentiation and augmentation is limited. A second approach is to develop for a platform of a certain configuration. For example, the host device must possess an interface of $\frac{1}{4}$ VGA, and support the rendering of MP3. This is invariably a higher spec platform and will only be available to a subsection of the customer base. However, it does allow for the provision of high quality, high revenue generating services. For B2B and B2E solutions, this strategy is attractive. In the case of B2C, the case is less clear cut, with the application domain and profile of the end-user being key determinants.

From a software engineering perspective, harnessing internet technologies offers a number of advantages. Internet Protocol (IP) is supported by practically all network operators, irrespective of their deployed networking technology. Many phones have micro browsers that enable pages constructed in HTML, or some variant, be viewed. In the case of multimedia, SMIL [4] viewers have been ported to mobile devices. In essence, if a service can be implemented using the rich array of internet technologies available, even those that may have to be simplified for mobile devices, the development process is simplified. However, this is not always feasible or desirable.

Service composition may demand the sourcing of data or use of other services from disparate sources. This is another source of heterogeneity in the mobile service provision process. Such services may use a series of highly specialised protocols and data formats that may be incompatible with those adopted by the service provider, or more problematically, may have previously adopted a non-standardised approach. Either way, the problem of service integration for composite service provision remains. Potential solutions to the problem exist, but may require the expenditure of significant effort. For example, Simple Object Access Protocol (SOAP) [5], an implementation on XML, enables the exchange of structured information. Another approach that has been documented involves the use of agents as service wrappers. In

this case, legacy services would be encapsulated by an intelligent agent, and this agent would provide the service to other agents that constituted the service being provided.

2.4 Why Agents?

Agents represent a paradigm for modelling and implementing software. Their characteristics are reasonably well defined but how these are harnessed is at the discretion of the software engineer. The degree to which an agent may be considered intelligent is open to interpretation, and implementation. Those in the AI community would envisage agents as encompassing a sophisticated reasoning engine. Others might view agents as a series of collaborating entities or even software objects. Irrespective of one's viewpoint, agents have been harnessed in a multitude of domains including those pertinent to this discussion, namely enterprise information systems and mobile computing.

Many researchers have demonstrated the applicability of the agent paradigm in various aspects of enterprises information systems. Kishore et al [6] investigate the agent paradigm as a means of realising integrated business information systems. Nahm & Ishikawa [7] adopt a hybrid multi-agent approach for inter-enterprise function, resource integration and collaboration in networked manufacturing systems. Agents have been harnessed for workflow management as they are seen as providing a robust and open solution to deliver dynamic and automatic workflow adaptations – a necessity in modern business environments [8][9]. In a similar vein, BDI agents have been adopted for realizing agile business processes [10]. In some cases, internet technologies have been harnessed, For example Li [11], describes AgentStra, an internet agent based system for strategic decision making. The potential of agents for Customer Relationship (CRM) management has been investigated [12], as has their applicability in WWW services [13].

While there are documented mobile computing systems that interacted with agent-enabled services, those that incorporate agents on the mobile devices are rare, not surprisingly. The mobile tourist domain has been the focus of two pioneering efforts. CRUMPET [14] harnesses agents to monitor the QoS of the wireless connection. In addition, it utilizes legacy services via agent wrappers. Gulliver's Genie [15] is concerned with the timely delivery of multimedia services to tourists, and uses agents to capture and interpret the tourist's contextual state. Ad-me [16] captures end-user biometric feedback to identify opportunities to deliver advertisements to users. EasiShop [17] encompasses a suite of agents, some mobile, to bargain with those traders that the shopper is in the vicinity of.

2.5 Agents: Integrating Enterprises with Mobile Subscribers

Deploying agents on mobile devices offers one potential avenue for enterprises to provide services to mobile users, be they customers or employees. From a software modelling perspective, the MAS may encompass companies' fixed networked server nodes and well as mobile nodes. This offers a standardised method for modelling the various components and the interactions between them. While agents offers some

advantages, such as managing the heterogeneity discussed previously, it is the enablement of a single platform for service deployment that is of most importance, and offers most potential. The key factor in this has been the development of new agent frameworks that can operate on mobile devices. The availability of a common software platform across many devices, namely J2ME, is also a key contributor.

In the next section, the software engineering cycle is considered from an agent perspective, concluding with a brief discussion on agent frameworks for mobile devices

3 Agent Oriented Software Engineering

Software engineering of agent systems follows the same steps as traditional software engineering. One common concern about agents would naturally concern their incorporation into methodologies for software development. To meet this, a number of researchers have considered various stages of the software engineering cycle and proposed methodologies and frameworks for each from an agent perspective. As an illustration of this, some proposed approaches in the various stages are now considered.

3.1 AOSE: Requirements Engineering

TROPOS [18] covers five phases of the software development cycle including Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation. It is particularly concerned with addressing a perceived lack of support for mentalistic or knowledge level notions in other proposed methodologies. Recall that the traditional goal of a software designer is to achieve a sufficient level of abstraction from which classes, methods and other attributes can be easily identified. However, in doing this, one may lose sight of any mentalistic notions that were inherent in the original system design and these may need to be "resurrected" during implementation. Thus TROPOS sets itself the goal of supporting knowledge level specifications throughout the development cycle. AUML [19] diagrams are adapted for the design phase. TROPOS has been illustrated using a sample e-commerce application, which was developed using JACK [26].

3.2 AOSE: Design Methodologies

Prometheus

Prometheus [20] was motivated by a desire to develop a process that could be taught to industry practitioners and students who might not have a background in intelligent agents. In particular, it is concerned with the development of agents of a BDI-like genre. While conceding that this makes the methodology less attractive to those working with other agent architectures, the authors were keen to address a perceived

deficiency in existing methodologies where agents were treated as simple interacting software processes, or as a black box. Prometheus supports three distinct though non-sequential development phases:

1. System Specification Phase

One of the first issues addressed is the identification and specification of how the agent system both perceives and interacts with its environment. In parallel, the high level objectives, termed functionalities of the system, must be identified. Other methodologies refer to these as roles. Finally, standard UML use case scenarios must be constructed.

2. Architectural Design Phase

The primary objective of this phase is the identification of the agents that will comprise the system. This is achieved by considering the functionalities identified during the specification phase and deciding how these might be optimally assigned to agents. After this, each agent is assigned a number of events to listen to and to react to. In addition, shared data objects are identified at this stage. Finally, a system overview diagram that lists the agents, events and shared data objects is constructed. This is arguably the most important artefact in the entire design process. Using this diagram as a basis, interaction diagrams and interaction protocols are constructed. Prometheus uses AUML notation for specifying interaction protocols.

3. Detailed Design Phase

During this phase the internal structure of each agent and how it will achieve its tasks in the system are specified. It is at this stage that Prometheus becomes specific to BDI-like agents and capabilities, internal events, plans and data structures for each agent are defined. These will provide the necessary details for implementation. Note that the detailed design will be heavily influenced by the choice of deployment platform.

SODA

SODA (Societies in Open and Distributed Agent spaces) [21] is a methodology for the development and design of internet-based Multi-Agent Systems. Most of the prevailing methodologies concentrate on intra-agent issues. In contrast, SODA concentrates on the inter-agent issues or the social aspects of Multi-Agent Systems. In addition, the agent's environment is of particular importance. SODA is based on the premise that agents cannot be considered in isolation from their proposed environment and advocates consideration of environmental issues from the very start. Therefore SODA is particularly concerned with the engineering of agent societies and their environment and does not concern itself with the actual agent specifics for which another methodology should be used. SODA supports the two standard phases of analysis and design.

1. Analysis Phase

After undertaking normal analysis tasks, for example, studying the application domain, identifying technology resources and constraints and so on, three models are constructed:

1. Role Model: the goals are modeled in terms of the tasks to be performed and these are then associated with roles and groups.
2. Resource Model: the environment is modeled in terms of the services available.
3. Interaction Model: All interactions are modeled in terms of interaction protocols and any interaction rules governing interaction within the system are identified.

2. Design Phase

During design, the abstract models produced during analysis are transformed into design abstractions that can be mapped on a one-to-one basis with actual components in the deployed system. Three models are produced:

- Agent Model: Individual and social roles are mapped onto agent classes.
- Society Model: Groups are mapped onto societies of agents. These are then designed and organized around a suitable co-ordination abstraction e.g. an Agent Communications Language (ACL).
- Environmental Model: Resources are mapped onto infrastructure classes and associated with topological abstractions e.g. domain names.

Agent Object Relationship (AOR) Modelling

Agent Object Relationship (AOR) [22] was proposed for the design of organisational information systems. Its *raison d'être* is simple: to remedy a deficit in traditional modelling approaches for handling both the dynamic and deontic aspects of organisational information systems. While AOR was inspired in part by Shoham's agent-oriented programming proposal [23], the methodology itself is heavily influenced by two popular database modelling methodologies, namely the Entity Relationship (ER) meta-model and the Relational Database (RDB) model. One of the objectives for AOR is to facilitate the transformation of an AOR model into an appropriate database schema in as simple a manner as possible. Clearly this implies the use of formal semantics when defining the elements of the AOR model.

In AOR, an entity can either be an agent, an object, an event, an action, a claim or a commitment. AOR is quite specific about the difference between agent and objects. In AOR, an object is a passive entity, a definition that contrasts with much of the language used in the literature. Only agents can perceive events, perform actions, communicate or make commitments. AOR assumes agents communicate via an Agent Communication Language (ACL) and uses a mentalistic model of knowledge representation. It also considers the concepts of perception, for example, of the environment, and memory, for example, of past events and actions, quite important.

3.3 AOSE: Implementation Frameworks

Agent Factory

Agent Factory [24] provides a cohesive framework for the development and deployment of agent-based applications. In particular, it provides extensive support for agents of the BDI genre. The Agent Factory runtime environment includes a Message Transport System which manages the delivery of messages between agents on the same platform and between agents on different platforms. A Migration Manager oversees the migration of agents between different platforms. A dedicated agent, the White Pages Agent, supervises the agent platform and manages the creation and deletion of agents. Occupying a layer above the agent runtime environment is the agent development environment. This contains a number of tools for enabling the fabrication of agents as well as providing features for monitoring and dynamically configuring various aspects of an agent's behaviour.

Jade

JADE (Java Agent Development Framework) [25] is a FIPA compliant agent platform that facilitates agent development by providing a set of agent-related services. Conceptually, it may be regarded as a kind of agent "middleware" solution. The platform itself can be distributed across several hosts, the only requirement being that a JVM is available on each. JADE does not subscribe to any particular agent theory or architecture and leaves the implementation details to the developer. However, a general agent model has been developed that can be easily customised to an architecture of the designer's choosing.

Jack

The JACK Intelligent Agent™ framework [26] was developed by Agent Oriented Software Pty. in Australia with the aim of bringing the intelligent agent concept into mainstream commercial software engineering as well as Java environments. JACK is designed primarily for use as a component in a larger software environment as this is seen as the most appropriate way of incorporating it with legacy code. Therefore a JACK agent is regarded as just another object by non-agent software. Conversely, the agent is equipped with a number of mechanisms for easily interfacing with other components.

3.4 AOSE: Frameworks for Embedded agents

LEAP

LEAP (Lightweight and Extensible Agent Platform) [27], originally an EU's Information Society Technologies (IST) project, is a FIPA compliant platform that can be used to deploy agents on a heterogeneous network of fixed and mobile devices. LEAP has evolved from the JADE platform and can interact with legacy JADE agents. The LEAP platform itself is modular and consists of the following modules:

- Kernel Module: This module is mandatory as it manages the platform and enables the realization of the agents.
- Communications Module: A mandatory module for handling the heterogeneous communications protocols.
- Administration module: An optional module that contains a GUI for managing and visualising agents.

AFME

AFME (Agent Factory Micro Edition) [28] was specifically designed for deploying intentional agents on mobile devices. It is based on Agent Factory and can interoperate with it. Designed for mobile phones, it requires the availability of the Constrained Limited Device Configuration (CLDC) Java platform augmented with the Mobile Information Device Profile (MIDP). AFME agents follow a sense-deliberate-act cycle. A perceptor feature is used to monitor the state of the agent's environment. A belief resolution function enables the identification of what it is that the agent can achieve within the current cycle. Appropriate actuators are then harnessed to realize the necessary change within the environment.

4 EasyLife

Fig. 1. Overview of EasyLife

EasyLife [29] provides a framework that enables service providers to deliver customised combinations of location-aware services to their subscribers. Many

examples of location-aware services have been documented, for example, AccesSights[30], HIPS [31], CoMPASS [32] and ACCESS [33]. However, EasyLife, like ACCESS, adopts an agent-based approach, and is concerned with service fusion through a suite of heterogeneous agents.

4.1 Overview of Easy Life

EasyLife is a multi-tier, location-aware, service-oriented system (Figure 1). It uses agents for realizing different services. At present, it consists of three services, a weather service (location-aware weather using the REST API and a mashup for the Yahoo weather services), a restaurant service (a location-aware recommending system) and a shopping service (a location-aware advertising system). With these services, users can instantly get the weather forecast for their location. Users can also determine where the nearest restaurant is. And enterprises can promote their services or products by delivering their advertisements to those people within a certain range. EasyLife harnesses GPS, Bluetooth, J2ME and AFME for its mobile client. It harnesses Wi-Fi as the wireless carrier for communicating with the service host. On the server side, it harnesses servlets, Jade, Hibernate and MySql for service delivery. EasyLife is modeled on a suite of heterogeneous agents that collaborate to deliver the required service.

4.1.1 Characteristics of Easy Life

Three characteristics define EasyLife:

1. EasyLife is truly a Software-as-a-Service. Customers may easily subscribe and use the services in an on-demand fashion. Service providers can focus on service development and service access.
2. EasyLife is more than just an ad-hoc service delivery application, providing a framework for building other customized services. It is a generic composite application that enables easy integration of services – either those provided by a third-party or ones based on core enterprise functions.
3. EasyLife is extensible and scalable. Its agent-based architecture is essential for realizing these features.

4.1.2 EasyLife services

EasyLife implements three simple but useful services, all using location as the primary context, and leveraging Web 2.0 service by the mobile client (Figure 2). Some scenarios for using EasyLife are as follows:

- **Scenario 1:** A subscriber is planning on playing some outdoor sports, and would like to know the weather forecast. By selecting the weather service, the subscriber could instantly obtain short and longer term forecasts.
- **Scenario 2:** While shopping, the subscriber would like the location of the nearest restaurant. By selecting the restaurant service, the corresponding agent-based service finds the nearest restaurant registered with EasyLife and informs the subscriber of its location. Should this be unsatisfactory, other nearby restaurants will be recommended.

- **Scenario 3:** While shopping on a busy street, those shops registered with EasyLife may send advertisements and details of special offers to the subscriber, but only if in the vicinity of the shop.

Fig. 2. Use Case scenario for EasyLife

4.1.3 Accessing EasyLife Services

Figure 3 illustrates the workflow for EasyLife services. After a service has been selected, the gateway agent is informed via a message encoded with the location and required service. This agent forwards the message to the income controller agent, which parses the message for location and desired service, and forwards the request to the appropriate service agent. The service agent processes the request, and assembles a message for the subscriber. This in turn is forwarded to the outcome controller agent and then to the gateway agent for dispatch to the mobile client. The message will contain some combination of text, multimedia, or a URL, and will be rendered on the mobile device for viewing.

4.2 Architecture of EasyLife

From an architectural perspective, EasyLife consists of three key components :

- EasyLife client;
- AgentProxy_app;

- AgentServices.

The benefits for dividing the system into these three different parts are that the resultant system is loosely coupled and each component is well abstracted and easily extended. Each component is now briefly described.

Fig. 3. Workflow for an EasyLife service

4.2.1 EasyLife Client

An illustration of the architecture of the EasyLife client may be seen in Figure 4. It is composed of five key elements:

- **Agent:** controls the behavior and workflow of the EasyLife client. It also listens for and handles all generated events.

Fig. 4. EasyLife – Client Architecture.

- **Context:** is responsible for obtaining the context from the external physical environment. In particular, it gets the location information from the external Bluetooth GPS receiver. It can be extended to handle other sensors if need be.
- **Controller:** provides an access point for certain functions – one example being the **UIController**. It provides an API for calling different UIs. For extensibility purpose, other controllers could be developed.
- **UIs:** provides the Graphic User Interface (GUI) for the EasyLife client.
- **Data Model:** represents an abstraction of the data from different domains, such as the GPS data model, the weather data model and so on.

Fig. 5. AgentProxy_app architecture.

4.2.2 AgentProxy_app

The AgentProxy_app is best viewed as a virtual router. It has a Gateway Agent which parses service requests, and forwards them to the correspondent agent service. It also relays messages to the mobile client. From an implementation perspective, agentProxy_app is implemented as a servlet (Figure 5). It communicates with the EasyLife client via WiFi, and standard TCP/IP with the Gateway Agents which acts as the interface to the Agent Services.

4.2.3 AgentServices

AgentServices consists of many different agents, each representing an enterprise and providing different services. There are controller agents which could be seen as service brokers interacting with other systems. AgentServices consists of two parts, controller agents and service agents (Figure 6).

Controller Agents

Income Controller Agents are the entry points for the agent service. They are responsible for parsing the request and dispatching it to the correspondent agent according to the location and the type of the service. They also inform the Outcome Controller Agent of the address of the Gateway Agent, so that the Outcome Controller Agent knows where to send the service data.

Outcome Controller Agents obtains the address of the Gateway Agent from the Income Controller Agent. All the service agents send their data to the outcome agent and this dispatches the service data to the correspondent Gateway Agent.

Service Agents

In general, all the service agents perform a similar role including the following functions:

- Event Listening;
- Service Data Constructing;
- Sending Messages.

In the case of the Restaurant service, four agents were constructed - each representing different restaurants, namely the KFC Agent, the PizzaHut Agent, the BurgerKing Agent and the Charlies Agent. The restaurant service demonstrates the location-based recommending service.

Four agents for the Shop service were created, representing different shops. The Shop service demonstrates a location-based advertisement service.

One agent for the weather service was created. This leverages the Yahoo weather service by calling two REST APIs - one for getting the code for the location, the other for getting weather forecast information from the Yahoo website.

Fig. 6. AgentServices architecture.

4.3 Implementation

EasyLife harnesses a number of technologies but intelligent agents from the key constituent components. Two different agent development environments are used – JADE for the resource intensive server components and AFME for the lightweight mobile devices. The EasyLife client is implemented using CLDC 1.1 and MIDP 2.0, as most mobile phones currently available in the marketplace supports these standards. Figure 7 illustrates the class hierarchy for EasyLife client. The servlet engine is deployed using Tomcat. The server agents are all implemented in JADE; databases are implemented in MySQL and Hibernate for object persistence. A summary of the technologies used is as follows:

EasyLife client

- JSR-139: Connected Limited Device Configuration 1.1

- JSR-118: Mobile Information Device Profile 2.0
- JSR-82: Java APIs for Bluetooth technology 1.0
- JSR-135: Mobile Media API 1.0
- AFME: Agent Factory Mobile Edition

AgentProxy_app

- Servlet 2.5: Tomcat runtime environment
- JADE 3.5: Java Agent Development Framework

AgentServices

- JADE 3.5: Java Agent Development Framework
- Hibernate 3.2: O/R mapping
- Mysq-connector-java-5.0.7: JDBC connector

Fig. 7. EasyLife Client Class Diagram

4.4 The EasyLife User Interface

Fig. 8. EasyLife Welcome screen (a) and service options (b).

EasyLife has been deployed on a Nokia N91 platform. Figure 8a illustrates the EasyLife welcome screen. The subscriber must then select the required service from the three standard EasyLife services (Figure 8b).

Fig. 9. Example of the EasyLife weather service (a) and the restaurant locator (b)

Figure 9a depicts what an archetypical weather service report looks like. The restaurant locator service identifies the nearest restaurant (Figure 9b).

5 Conclusion

EasyLife demonstrates the viability of the agent paradigm for integrating heterogeneous information sources for service provision to mobile users. The robustness of the agent platform is verified through the use of two separate agent toolkits being harnessed to deliver EasyLife services. Developments in Agent-Oriented Software Engineering (AOSE) and agent frameworks are resulting in an increasing range of options for those considering harnessing the characteristics of agents for their applications and services. The more recent development of agent platforms for mobile devices represents a new frontier for Multi-agent Systems. From an enterprise perspective, it validates MASs as a candidate solution of extending the reach of legacy services to the mobile computing community, for combining existing services in new and innovative ways or indeed, delivering new original services to mobile users.

Acknowledgments. This material is based upon works supported by the Science Foundation Ireland (SFI) under Grant No. 07/CE/I1147

References

1. Vasilakos, A., Pedrycz, W. (eds.): Ambient Intelligence, Wireless Networking, Ubiquitous Computing, Artec House. (2006)
2. Hook, K.: Steps to take before intelligent user interfaces become real, *Interacting with Computers*, 12 (4), (2000) 409-426
3. Doctor, F., Hagras, H., Callaghan, V.: A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments, *IEEE Transactions on Systems, Man and Cybernetics Part A*, 35(1) (2005) 55-65
4. Ayars, J., Bulterman, D., Cohen, A., Day, K., Hodge, E., Hoschka, P., Hyche, E., Jourdan, M., Kim, M., Kubota, K., Lanphier, R., Layaida, N., Michel, T., Newman, D., van Ossenbruggen, J., Rutledge, L., Saccocio, B., Schmitz, P., ten Kate, W., Michel, T.: Synchronized multimedia integration language (SMIL 2.0) Second edition. W3C recommendation (2005)
5. Louridas, P.: SOAP and Web Services, *IEEE Software*, 23(6) (2006) 62-67
6. Kishore, R., Zhang, H., Ramesh, R.: Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems, *Decision Support Systems*, 42 (1) (2006) 48-78
7. Nahm, Y.-E., Ishikawa, H.: A hybrid multi-agent system architecture for enterprise integration using computer networks, *Robotics and Computer-Integrated Manufacturing*, 21 (3) (2005) 217-234
8. Muller, R., Greiner, U., Rahm, E.: AW: A workflow system supporting rule-based workflow adaptation, *Data & Knowledge Engineering*, 51 (2) (2004) 223-256

9. Ehrler, L., Fleurke, M., Purvis, M.A., Savarimuthu, B.T.R.: Agent-Based Workflow Management Systems (WfMSs): JBees - A Distributed and Adaptive WFMS with Monitoring and Controlling Capabilities. *Information Systems and E-Business Management* 4(1), 5–23 (2006)
10. Burmeister, B., Arnold, M., Copaciu, F., Rimassa, G.: BDI-Agents for Agile Goal-Oriented Business Processes, Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Estoril, Portugal, (2008) 37-44
11. Li, S., AgentStra: an Internet-based multi-agent intelligent system for strategic decision-making, *Expert Systems with Applications*, 33 (3) (2007) 565-571
12. Pani, A.K., Venugopal, P.: Implementing e-CRM using intelligent agents on the internet, *International Conference on Service Systems and Service Management*, (2008) 1-6
13. Dickinson, I., Wooldridge, M.: Agents are not (just) Web Services: Considering BDI Agents and Web Services. In: *Proc. of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005)*, Utrecht, The Netherlands (2005)
14. Poslad, S., Laamanen H., Malaka R., Nick A., Zipf, A.: Crumppet: Creation of user-friendly mobile services personalized for tourism, *Proceeding of the Second IEE International Conference on 3G Mobile Communication Technologies*, London, UK. (2001)
15. O'Grady, M.J., O'Hare, G.M.P., Sas, C.: Mobile Agents for Mobile Tourists: A User Evaluation of Gulliver's Genie, *Interacting with Computers*, 17 (4) (2005) 343-366
16. Hristova, N., O'Hare, G.: Ad-me: Wireless Advertising Adapted to the User Location, Device and Emotions. *Proceedings of the 37th Hawaii International Conference on System Sciences*, Hawaii, (2004)
17. Keegan, S. O'Hare, G.M.P., O'Grady, M.J.: EasiShop: Ambient Intelligence Assists Everyday Shopping, *Journal of Information Sciences Elsevier Press*, 178 (3) 588-611
18. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Modeling early requirements in Tropos: a transformation based approach. In: Wooldridge, M., Ciancarini, P., Weiss, G. (eds.): *Agent-Oriented Software Engineering II. Lecture Notes in Computer Science*, vol. 2222. Springer-Verlag, Berlin Heidelberg New York (2000) 151–168
19. Odell, J., Van Dyke Parunak, H., Bauer, B.: Extending UML for Agents, *Proceedings of the Agent-Oriented Information System Workshop at the 17th National Conference on Artificial Intelligence*, Austin, USA. (2000) 3-17
20. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents, *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, Bologna, Italy. (2002)
21. Omicini, A.: SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In: Ciancarini, P., Wooldridge, M.J. (eds.): *Agent-Oriented Software Engineering, Lecture Notes in Computer Science*, vol. 1957. Springer-Verlag, Berlin Heidelberg New York (2000) 185–193
22. Wagner, G.: The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. *Information Systems*, 28:5 (2003) 475 - 504
23. Shoham, Y.: Agent-oriented Programming. *Artificial Intelligence*, 60(1) (1993) 51–92
24. Collier, R., O'Hare, G., Lowen, T., Rooney, C.: Beyond prototyping in the factory of the agents. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.): *Proceedings of the 3rd Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03)*, *Lecture Notes in Artificial Intelligence*, vol. 2691. Springer-Verlag, Berlin Heidelberg New York (2003) 383–393
25. Bellifemine, F., Caire, G., Greenwood, D.: *Developing Multi-Agent Systems with JADE*. New Jersey: John Wiley & Sons. (2007)
26. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical Report, Agent Oriented Software Pty. Ltd, Melbourne, Australia (1998)

27. Bergenti, F., Poggi, A.: LEAP: A FIPA Platform for Handheld and Mobile Devices. In Proc. Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), Seattle, CA (2001) 303–313
28. Muldoon, C., O'Hare, G.M.P., Collier, R. O'Grady, M.J.: Agent Factory Micro Edition: A Framework for Ambient Applications. In: Proceedings of Intelligent Agents in Computing Systems (IACS 2) Workshop held in Conjunction with International Conference on Computational Science (ICCS) 2006 Reading, UK. Lecture Notes in Computer Science (LNCS), vol. 3. Springer-Verlag, Berlin Heidelberg New York (2006) 727-734
29. Shen, J.M., O'Grady, M.J., O'Hare, G.M.P.: EasyLife: A Location-aware Service Oriented Mobile Information System. Proceedings of 12th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES2008), Zagreb, Croatia. Lecture Notes in Artificial Intelligence (LNAI), vol. 5177. Springer-Verlag, Berlin Heidelberg New York (2008) 229-236
30. Klante P, Krösche J, Boll S.: AccessSights - a multimodal location-aware mobile tourist information system, Proceedings of the 9th International Conference on Computers Helping People with Special Needs (ICCHP). (2004)
31. O'Grady, M.J., O'Rafferty, R.P., O'Hare, G.M.P.: A tourist-centric mechanism for interacting with the environment. In: Proceedings of the First International Workshop on Managing Interactions in Smart Environments, Dublin, Ireland, Springer (1999) 56–67
32. Weakliam, J. Lynch, D., Doyle, J., Bertolotto, M., Wilson, D. Delivering Personalized Context-Aware Spatial Information to Mobile Devices. Proceedings of the 5th International Workshop on Web and Wireless Geographical Information Systems, 2005, Lausanne, Switzerland. Lecture Notes in Computer Science (LNCS), vol. 3833. Springer-Verlag (2005) 194-205
33. Strahan, R., O'Hare, G.M.P., Phelan, D., Muldoon, C., Collier, R.: ACCESS: An Agent based Architecture for the Rapid Prototyping of Location Aware Services. In: Proceedings of the 5th International Conference on Computational Science (ICCS 2005), Emory University Atlanta, USA. (2005)