applied sciences

MDPI

Article

Linked Data Aware Agent Development Framework for Mobile Devices

İlker Semih Boztepe and Rıza Cenk Erdur *

Department of Computer Engineering, Ege University, 35100 Bornova, İzmir, Turkey; ilker.semih.boztepe@ege.edu.tr

* Correspondence: cenk.erdur@ege.edu.tr; Tel.: +90-232-311-2597

Received: 28 August 2018; Accepted: 1 October 2018; Published: 6 October 2018

Abstract: Due to advances in mobile device and wireless networking technologies, it has already been possible to transfer agent technology into mobile computing environments. In this paper, we introduce the Linked Data Aware Agent Development Framework for Mobile Devices (LDAF-M), which is an agent development framework that supports the development of linked data aware agents that run on mobile devices. Linked data, which is the realization of the semantic web vision, refers to a set of best practices for publishing, interconnecting and consuming structured data on the web. An agent developed using LDAF-M has the ability to obtain data from the linked data environment and internalize the gathered data as its beliefs in its belief base. Besides linked data support, LDAF-M has also other prominent features which are its peer-to-peer based communication infrastructure, compliancy with Foundation for Intelligent Physical Agents (FIPA) standards and support for the Belief Desire Intention (BDI) model of agency in mobile device agents. To demonstrate use of LDAF-M, an agent based auction application has been developed as a case study. On the other hand, LDAF-M can be used in any scenario where systems consisting of agents in mobile devices are to be developed. There is a close relationship between agents and linked data, since agents are considered as the autonomous computing entities that will process data in the linked data environment. However, not much work has been conducted on connecting these two related technologies. LDAF-M aims to contribute to the establishment of the connections between agents and the linked data environment by introducing a framework for developing linked data aware agents.

Keywords: agent development framework; mobile device agent; linked data; semantic web

1. Introduction

An agent is a computational entity that is capable of autonomous action on behalf of its users in order to satisfy its delegated objectives [1]. Autonomy is generally considered as an essential property of an agent. On the other hand, Wooldridge and Jennings suggest three other additional properties that an intelligent agent is expected to have [2]. These properties are reactivity, proactivity, and social ability. While reactivity addresses the ability of an agent to perceive its environment and respond to changes that occur in its environment, proactivity is related with exhibiting goal-oriented behavior. Finally, social ability is the capability of an agent for interacting with other agents and possibly human users by exchanging messages represented in a specific agent communication language. Agents have found many application areas in various domains such as human computer interaction (e.g., digital personal assistants), information retrieval, electronic commerce, and manufacturing since the mid-1990s [3,4]. Recent studies show that use of agent technology is still in progress in various domains such as cyber physical systems and internet of things that are popular research areas in the context of industry 4.0 [5,6].

Architecture of an agent typically consists of two layers. The first one is the communication layer, which provides an agent with the necessary means for communicating with other agents by exchanging agent communication language messages over a network infrastructure. The second one is the agency layer, where deliberation and planning mechanisms of an agent are implemented. In other words, properties of reactivity and proactivity are provided in the agency layer. It would of course be impractical to develop an agent by implementing these layers from scratch each time a new agent based application is being developed. Thus, agent development frameworks have been introduced to ease and expedite the implementation of agent based solutions. JADE [7], JADEX [8], and Jason [9] are the most widely known open source agent development frameworks. Interested readers can refer to Reference [10] which is a comprehensive survey of agent development frameworks.

Due to advances in wireless network and hardware technologies, mobile devices such as mobile phones and tablets, through which people can access information from anywhere and anytime, have already become an indispensable part of our lives. This situation has inevitably affected the agent research community and studies have been conducted on deploying agent development frameworks into mobile devices. As a result of these studies, several agent development frameworks that address mobile devices have been released. Jade LEAP [11], 3APL-M [12], and Agent Factory Micro Edition (AFME) [13] are examples of such frameworks. As will be discussed in the related work section, each of these frameworks has its own prominent features. For example, while Jade LEAP is distinguished by the agent communication and execution infrastructure that it provides, 3APL-M and AFME employ deliberation mechanisms which are based on the Belief-Desire-Intention (BDI) model of agency [14] for developing intelligent agents. Recently, JADE and JADEX released new versions that support the Android Operating system [15,16].

Another technology that has close relationship with the agent technology is the semantic web. Since the beginning of the semantic web movement, agents have been considered as the main type of software that would consume the data on the semantic web. In their original article about the semantic web, Berners-Lee, Hendler and Lassila [17] emphasized the connection between agents and semantic web by defining agents as the autonomous computing entities that would be used in the semantic web environment. Linked data, which presents a set of techniques and standards to publish, link and query data on the web, is considered the realization of the semantic web vision, since it had a great impact on the development of real-life semantic web applications [18,19]. Thus, the number of real-life semantic web applications has begun to increase with the introduction of linked data.

On the other hand, although the close relationship has been emphasized from the very beginning, in practice there has been little collaboration between agent and semantic web/linked data communities.

In this paper, we introduce the agent development framework that we have developed for mobile devices. Using this framework, developers can build intelligent agents that have the attributes of social ability, reactivity, proactivity, and autonomy. The main contribution of the presented framework is its linked data support. Linked data support in an agent simply corresponds to the ability to supply its beliefs from the linked data environment, to keep the obtained beliefs internally, and use those beliefs during planning process.

By providing linked data support, we also aim to contribute to the establishment of the links between agents and the linked data environment. Since the ability to supply its beliefs from the linked data environment is the developed agent framework's distinguishing feature, we called it Linked Data Aware Agent Development Framework for Mobile Devices (LDAF-M). The belief component of LDAF-M uses the Resource Description Framework (RDF) model to represent the beliefs of an agent and provides a graph data structure to hold the beliefs internally. By this way, semantic data stored in ontologies as well as in open data stores, such as Freebase or DBpedia in the linked data environment, can be used to represent an agent's beliefs. Linked data support, which is the distinguishing property of LDAF-M, makes it the first linked data aware agent framework for mobile devices to the best of our knowledge.

LDAF-M agents can be qualified as intelligent agents according to the definition of Wooldridge [2], since they support the attributes of autonomy, reactivity, proactivity, and social ability. Social ability is related to the ability of an agent to exchange agent communication language messages with others in the system. Social ability is provided through the LDAF-M communication infrastructure. Reactivity is the ability to respond to the changes in the environment, and proactivity is related to exhibiting goal directed behavior. Reactivity and proactivity are satisfied through the use of the BDI model of agency [14]. Finally, autonomy is satisfied through the use of a planner within the context of the BDI architecture. Communication infrastructure and the agency layer are discussed in detail in Section 2, where the architecture of LDAF-M is given. On the other hand, some highlights related to the communication infrastructure, standards followed during the development, and BDI model of agency are given below.

Support for peer-to-peer (P2P) paradigm: In the communication layer, LDAF-M provides a P2P based communication and resource sharing infrastructure where each mobile device behaves as a peer. The P2P infrastructure provided by LDAF-M also includes a mechanism that can detect a peer that becomes offline due to a problem and can handle this situation so that the system continues to operate seamlessly.

FIPA compliant execution platform: FIPA (Foundation for Intelligent Physical Agents) is an IEEE Computer Society standards organization that issues a set of specifications which intend to promote interoperability among heterogeneous agent systems [20]. LDAF-M provides a FIPA compliant platform for mobile device agents. Jade LEAP and AFME also support FIPA standards. On the other hand, during the design of LDAF-M, we have made slight modifications which are discussed in Section 2.2.

Support for BDI model of agency: Belief-Desire-Intention (BDI) model of agency [14,21,22] is a framework for describing the behavior of agents in terms of their beliefs, desires (options) and intentions (adopted options). This model of agency draws its inspiration from practical reasoning which is a model of decision making proposed by Bratman [23]. The agency layer of LDAF-M provides the necessary infrastructure to develop BDI agents for mobile devices. 3APL-M and AFME also support BDI model of agency. However, different from those frameworks, LDAF-M has the ability of representing beliefs of an agent using semantic web standards (i.e., RDF) as mentioned before.

The rest of the paper is organized as follows. The layered architecture of LDAF-M is explained in Section 2. P2P communication 1. Section 2.2, first overviews FIPA standards and then discusses compliancy of LDAF-M to the FIPA standards. In Section 2.3; the agency layer, where the BDI model is implemented, is explained. Linked data support is discussed in Section 3. Section 4, which is the case study and discussion section, consists of three parts. The first part illustrates how LDAF-M is used in developing agent systems. In the second part, an agent based auction application that has been developed using LDAF-M is introduced by giving a detailed interaction diagram. In the third part, the case study given is discussed. Related work and conclusion are given in Sections 5 and 6 respectively. Finally, references are listed.

2. Architecture of LDAF-M

LDAF-M has a layered architecture which mainly consists of two layers, as shown in Figure 1. The bottom layer provides the communication infrastructure. LDAF-M also follows the FIPA specifications which mostly affect the communication and deployment infrastructure. FIPA compliancy is also explained in this section following the communication infrastructure. On top of the communication layer, the agency layer is located. The agency layer contains the implementation for the BDI model of agency.

The communication layer and the agency layer basically interact by means of agent communication layer messages. The agency layer extracts the objectives that it will try to satisfy by examining the incoming messages stored in the incoming message queue of the agent. On the other hand, as a result of executing a plan, some messages need to be sent to other agents. In this case,

the agency layer places the messages to be sent in the outgoing message queue, which then are sent to the related agents by the communication layer.

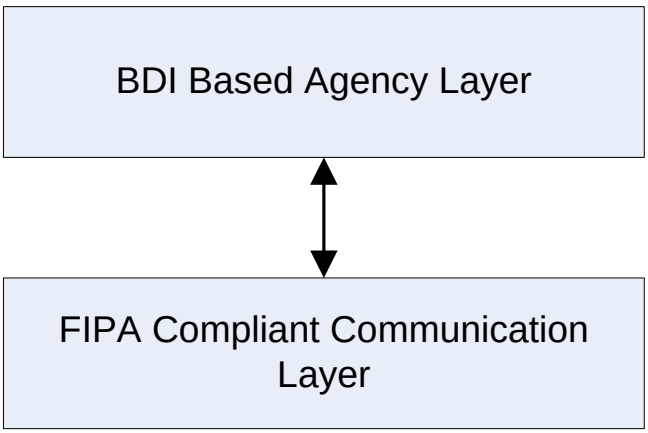

Figure 1. Linked Data Aware Agent Development Framework for Mobile Devices (LDAF-M) layers.

2.1. Communication Infrastructure

In this subsection, LDAF-M's communication mechanism, which has been developed following the P2P paradigm, is introduced. In addition, it is also discussed how the case of a peer that becomes inaccessible due to a problem arising from a resource constraint in a mobile device is handled in this P2P infrastructure.

P2P is an approach for network communication where shared data is processed by two or more client computers, each of which have equal responsibilities. In contrast to the client-server architecture where there are separate dedicated clients and servers, in a P2P infrastructure each of the peers can play the role of both a server and a client.

There are three approaches regarding the architecture of a P2P communication infrastructure [24–26]. These are central, decentralized, and hybrid approaches.

A dedicated central server and a group of peers are used in the implementation of a central P2P architecture. The central server maintains a database where it stores the indexes of all peers and the resources that belong to each peer. In the decentralized P2P architecture, there is not any central server that controls the network. Instead, each peer has equal functionality and holds information about its own resources. In other words, each of the peers acts as the index server. The hybrid P2P approach, which is the third-generation architecture, provides a hierarchical structure in which there are super peers having the properties of a central server.

Considering the constraints of mobile devices and the scale of the systems that would be constructed using LDAF-M, the communication infrastructure has been based on the centralized P2P approach. In the alternative (i.e., decentralized) P2P approach, since each peer keeps track of its own resources and needs to search resources of other peers, an extra load is introduced for peers. Additionally, a certain level of message traffic occurs on the network because of the circulating queries. On the other hand, using a centralized approach brings in several advantages such as easy implementation and timely responses to queries among peers. In addition, in our case, the connectivity status of a mobile device may dynamically change, as a result of running out of battery or going out of the range of the network. Using the centralized approach, we can easily keep track of and handle the connectivity problems, as it is explained in the following paragraphs.

As it can be seen from Figure 2, a group of peers and one central dedicated server have been used for the realization of the central P2P architecture. The dedicated server, which is called the "index server", is responsible for controlling and sustaining the P2P network. When any one of the peers becomes online, information associated with resources of this peer are recorded in the database within the index server. In the same way when any of the peers becomes offline, information associated

with that peer is removed from the database within the index server. In addition, each of the online peers sends notifications to the index server periodically in order to indicate that they still actively use the system. If the index server cannot receive notifications from a peer for a certain amount of time, the index server updates its database to indicate that this peer becomes offline.

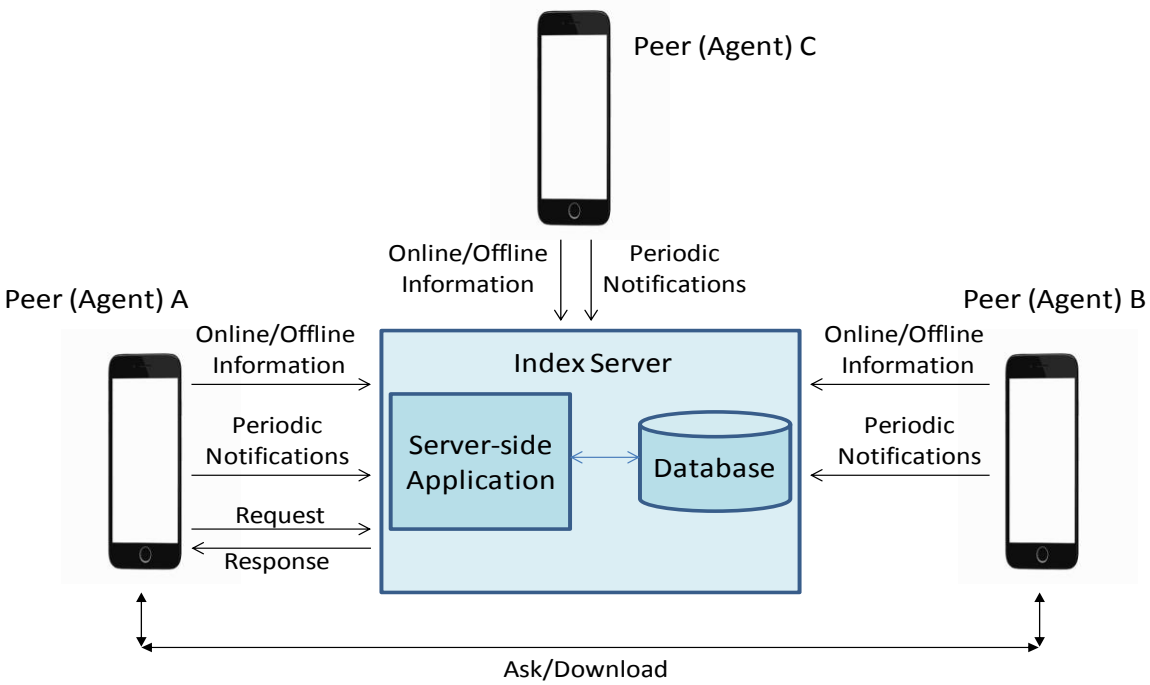

Figure 2. Peer-to-peer (P2P) communication infrastructure in LDAF-M.

While determining the length of the time interval for periodic notifications of the mobile device agents, we need to consider the timeliness of the notification, but at the same time we need to avoid introducing too much load for the network and the index server. A peer being offline may not be detected for a long time if periodic time intervals are too long. On the other hand, an extra load comes on the index server and the network when the periodic time intervals are too short. Therefore, the length of periodic time intervals was determined to minimize the negative effects of these two cases, which are opposite of each other. In particular, we have defined a specific notification time interval of 60 s based on our experiences in an agent based auction application that has been developed using LDAF-M.

The server-side application (e.g., Java Servlet based) on the index server may receive three main types of notification from a peer. The first type of notification is sent by a peer when it first becomes online. The second type of notification is sent by a peer when it decides to leave the system and will become offline. In both cases, the index server updates records in its local database according to the notification received. The third type of notification is sent as an acknowledgement that the peer is still actively participating in the system. This type of notification is sent periodically as explained above. When the index server realizes that a peer has not sent a notification for a long time, it understands that there is a problem with it and updates its local database to record the status of this peer as being offline.

Figure 2 also shows the working mechanism of the central P2P approach. The peer labeled as "Peer A" sends a request to the index server. Upon receiving this request, the index server queries its local database and sends the obtained result to "Peer A". This result includes information about the peers that can provide the information in response to the request of "Peer A". In Figure 2, it is assumed that the requested data is provided by the "Peer B". Consequently, the "Peer A" communicates directly with "Peer B" to obtain the requested data.

Both handling the notifications coming from the peers and behaving as the resource finder, the index server plays an important role in the communication of peers and provision of information and services for the whole platform.

2.2. FIPA Compliant Architecture

FIPA (Foundation for Intelligent Physical Agents) is an IEEE Computer Society standards organization that issues a set of specifications about agents and multi-agent systems. These specifications, which mainly aim to promote interoperability among agent systems, can be classified in terms of different categories. Agent communication, agent management, agent message transport and abstract architecture are the main categories of FIPA specifications [20].

During the design and implementation of LDAF-M, one of the specifications that we have followed to a great extend was the FIPA agent management specification [27]. FIPA agent management specification introduces the "agent management reference model", using which we could have mapped the central P2P communication infrastructure into a FIPA compliant architecture.

The agent management reference model introduced by the FIPA agent management specification [24] consists of the following components:

Directory Facilitator (DF), provides yellow page services to other agents. Agents register their services with the DF using a specific registration template or ontology and may query the DF to find out the required services offered by other agents.

Agent Management System (AMS), manages the life cycle of agents. Tasks such as creation, termination and suspension of agents are under the responsibility of the AMS. The AMS also keeps the Agent Identifiers (AIDs) which contain transport addresses of agents that registered with the platform. Initially, each agent registers with the AMS to get an AID for itself.

Message Transport Service (MTS), is responsible for delivering messages among agents residing on the same or remote agent platforms. Agents exchange messages expressed in an agent communication language and use transport addresses in delivering messages.

Agent Platform (AP), provides the physical environment on which agents can run properly. It consists of DF, AMS, MTS and agents.

The mapping made to a FIPA compliant architecture following the FIPA agent management reference model is shown in Figure 3. The index server and the peers (i.e., mobile devices), which are the main components of the central P2P communication infrastructure, have been implemented as different FIPA agent platforms, as shown in Figure 3.

Each of the agent platforms that reside on the mobile devices (peers) contains its own Agent Management System (AMS) as the managing authority in the agent platform. An AMS neither has any knowledge about other agent platforms, nor communicates with an AMS located in a different agent platform. Information about agents on remote platforms can be accessed through DF (Directory Facilitator) located in the agent platform that contains the index server. The tasks such as the creation of a new agent, handling the changes in the life cycle of an agent and the killing of an agent are under the responsibility of the AMS. When a new agent is created, Agent Identifier (AID) information that includes transport address(es) for that agent are recorded by the AMS. These records are kept within the AMS during the lifetime of the agent. More than one agent can run on an AP, and AIDs of the agents located within an AMS, can be queried by other agents which run on the same AMS.

The DF is in charge of providing yellow page services to all of the agents. Agents from different platforms can publish through the DF the services that they provide and these services are searchable from all of the mobile device agent platforms. Unlike similar frameworks such as Jade LEAP [11], LDAF-M contains only one DF and it is located in the agent platform that contains the index server. There does not exist an AMS in the index server's agent platform, rather there is an AMS in each mobile device agent platform. Since AMS manages life cycles and naming of agents in the system, it is obvious that placing a single AMS in the index server's agent platform may cause overloading of this single AMS. In LDAF-M, an agent looking for a specific agent first makes a search using the AMS in its

local platform. If it is understood that the specific agent being looked for is not in the same platform, then the search is extended to the DF residing in the index server's agent platform. With such a design, we aim to reduce the load on the index server's agent platform.

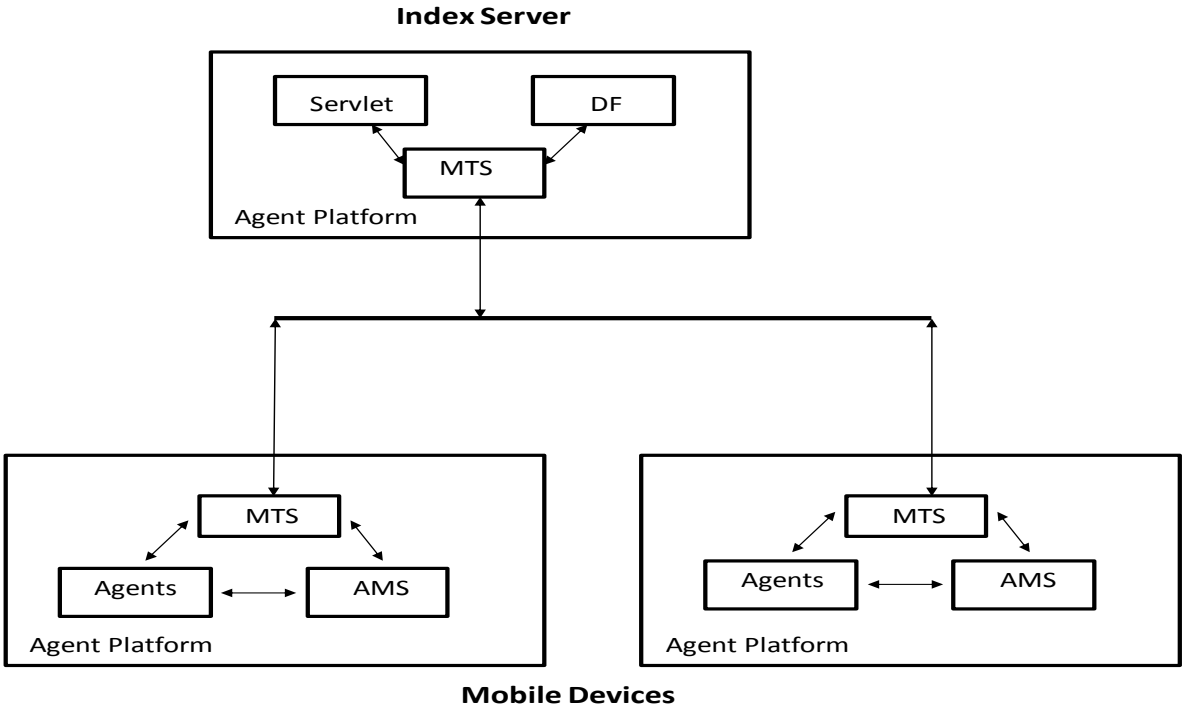

Figure 3. Mapping of the P2P communication infrastructure into Foundation for Intelligent Physical Agents (FIPA) compliant architecture.

As discussed in detail in the previous section, there is a notification mechanism between peers and the index server. By this way, peers that go offline because of battery or out of network range problems can be tracked. In such cases, the index server updates the entries in the DF to provide the most accurate information to agents located on mobile device agent platforms.

Message Transport Service (MTS), is the component that handles the message delivery between agents on the same or different platforms. The type of message transport protocol that is used in the communication between agents residing in a mobile device is the socket technology, which is actually on top of TCP/IP transmission protocol. On the other hand, the type of message transport protocol that is used in the communication between the index server and any mobile device is Hypertext Transfer Protocol (HTTP), since the agent platform running on the index server receives messages sent to it from the mobile devices using Java Servlet technology.

Agents exchange messages expressed in an agent communication language. FIPA also issued standards related with agent communication language and message structures. FIPA Agent Communication Language (FIPA-ACL) is the proposed agent communication language, and hence used in LDAF-M for the communication of agents. The structure of a standard FIPA message consists of two parts which are the message envelope and the message content. To overcome the memory limitations on the mobile devices, a slight modification has been performed on the structure of messages exchanged. This slight modification includes combining the envelope and the content parts of the messages to form a single deliverable unit and using only one of the optional message parameters that are similar. For example, the "to" and "from" parameters that can be used within the message envelope, and the "receiver" and "sender" parameters that are used in the content of a FIPA-ACL message have been combined into two single parameters.

Finally, agents running on the mobile device agent platforms are implemented as independent Java threads. Internally, each agent supports the agent life cycle standard proposed by FIPA. According

to this specification, an agent can be in one of five states, which are "initiated", "active", "suspended", "waiting", and "transit". Transitions among these states are also defined. When an agent is constructed, the finite state machine representing the life cycle begins at the initial state. The active state is the one where an agent continues to behave in order to meet its delegated objectives. In the active state, transport messages are transmitted to the agent in a normal fashion. Within the scope of suspended and waiting states, all activities of an agent are terminated temporarily. In suspended and waiting states, the message transport service stores the messages locally on behalf of the agent. After the agent passes to the active state again, the stored messages are delivered to it. In the transient stage, an agent is able to move to different platforms. In the current version of LDAF-M, agent mobility is not supported, since agents that travel between the platforms is a research topic in its entirety.

2.3. Agency Layer

The agency layer in LDAF-M framework is based on the Belief-Desire-Intention (BDI) architecture. BDI is a model of agency which has been widely used in the development of agents and multi-agent systems, including commercial agent software. Hence, the design and implementation of the basic modules in this layer are inevitably similar with the existing BDI supported agent frameworks. On the other hand, the main difference of LDAF-M is that it can obtain knowledge from linked data resources and handle the obtained knowledge. In this regard, new functionalities and internal constructs has been added to a standard BDI implementation. These variations are explained in Section 3, where linked data support is discussed.

The BDI architecture's origins lie in the theory of practical reasoning [23] developed by Michael Bratman. As stated in Reference [22], this kind of reasoning is always directed towards doing an action and consists of two activities. The first activity is deliberation which is about deciding what state of affairs the agent wants to achieve. The second activity is means-ends reasoning (planning) and it decides how to achieve those selected state of affairs using an available set of actions.

Practical reasoning is realized in agent systems based on the concepts of beliefs, desires and intentions. This is why these architectures are called BDI architectures.

Everything that an agent knows and believes about the world in which it is located, including itself and also other agents around it, is represented by the belief component within the BDI architecture. Agents obtain their beliefs by continuously perceiving the environment in which they operate.

The motivational stance of an agent is represented by the desire component located within the BDI agent architecture. The desire concept characterizes the goals of an agent which define what the agent wants to achieve. In other words, desires are the possible options of an agent [14,21,22,28].

At the end of the deliberation mechanism, desires which are adopted by an agent become intentions. This means that the adopted desire would lead to an action. In this regard, during the process of identifying the actions that an agent will perform, the intention component within the BDI agent architecture has a much stronger role than the desire component. An agent may host many desires which can possibly be transformed into intentions and then to actions in the near future. The agent evaluates desires in terms of agent's beliefs, other desires and intentions, and then suitable desires are adopted as intentions that would lead to actions.

In LDAF-M, beliefs may come from the environment in which the agent runs, or from the linked data environment. The messages delivered from other agents may also cause the beliefs of an agent to be updated. The ability to obtain its beliefs from the linked data environment (e.g., from interrelated RDF documents) is the distinguishing feature of LDAF-M. The beliefs are used in option generation together with the current intentions to produce desires. The filter function uses beliefs, desires and intentions to produce new intentions. The filter component has a typical rule base which is used in determining which goals to adopt. Primarily, the beliefs in the active state at the belief base are processed using this rule base. Then the filter component determines what goals to adopt by matching the beliefs in the active state and the rules defined in its own context. Since an intention

is directed towards an action, the related actions corresponding to an intention are executed next. Basic components of the BDI architecture that are implemented in LDAF-M are shown in Figure 4.

In fact, desire and intention which is a strong form of desire are similar notions, and in the implementation of BDI architectures they are represented by the "goal" concept. An agent may have many goals (i.e., desires) and the adopted goals are its intentions. The planning mechanism is the second stage of practical reasoning as mentioned above, and in order to transform an adopted goal into an action, the planning mechanism constructs a plan to fulfill that goal and executes the plan step by step.

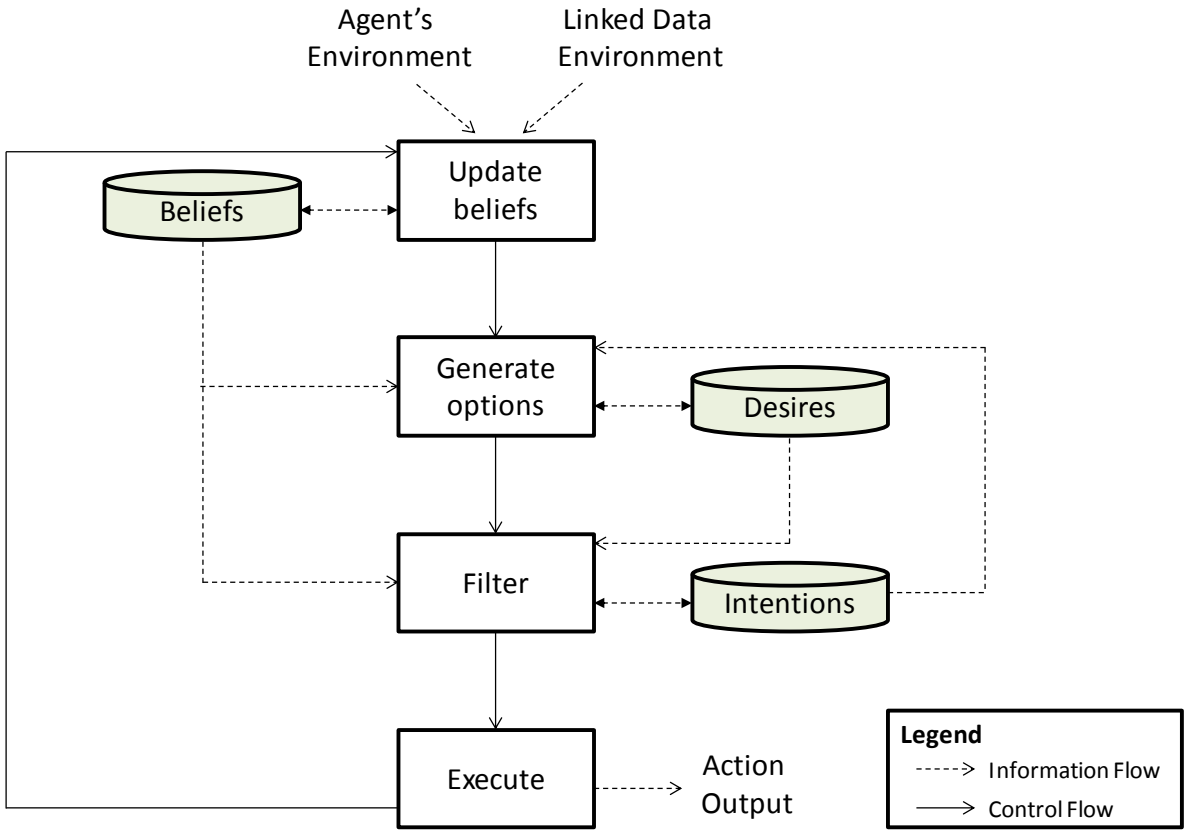

Figure 4. Control and information flow in the Belief-Desire-Intention (BDI) model based agency layer in LDAF-M.

Each plan consists of two parts. The header part includes information about which goal the plan is associated with and under what conditions it is applicable. The body part defines the actions to be performed or the sub-goals to be obtained. In this respect, there is a tight relationship between goals and plans, since a plan is executed after an agent's adoption of the goal to which that plan is related.

JADEX [8] is a well-known open source agent development framework that supports the development of BDI agents. The agency layer that we have implemented in LDAF-M was inspired by JADEX. On the other hand, since we target mobile devices, the agency layer that we have implemented works on a mobile device supporting the Java platform. As mentioned before, the other property that makes LDAF-M different from other BDI agent development frameworks, is the ability for collecting its beliefs from the linked data environment. Linked data support is discussed in the following section.

Both the agency layer and the communication layer of LDAF-M consist of several software packages that contain many classes. In the communication layer; platform, core, gui, AMS, DF, MTS, Message, MessagePattern, and MessageBase packages are included. These packages support agent platform operations on a mobile device, provide several graphical user interfaces that can be used in managing the platform, define the agent communication language messaging infrastructure, and

provide the FIPA compliancy. The agency layer consists of several software packages such as BeliefBase, GoalBase, PlanBase, Deliberation, LinkedData, DataModel, Conditions (drop, suspend, triggering conditions) that includes the classes necessary for implementing the BDI model of agency and linked data support. These packages have open source access. Please refer to GitHub page for the project [29].

3. Linked Data Support in LDAF-M

The linked data concept, which has been originally introduced by Tim Berners-Lee in his technical note [30], refers to best practices for publishing, interlinking and consuming machine processable data on the web. In practice, Resource Description Framework (RDF) is the data model used for publishing structured and machine processable data on the web. A dataset represented using RDF can also include links to other datasets. These links can then be followed to aggregate data from different datasets [31,32]. To perform queries on linked data, SPARQL query language [33] is used.

RDF uses triples to represent resources and their relationships. Each triple consists of a subject, predicate and an object. The subject is the resource being defined and hence is represented by a Uniform Resource Identifier (URI). On the other hand, the object in a triple may also have a Uniform Resource Identifier (URI) value, since it is possible for an object to identify other resources. Otherwise, the object part of the triple contains a literal value. The predicate plays a key role in determining the relationship of the subject and the object in the triple. Using URIs to identify resources makes it possible to represent RDF statements as a graph where nodes represent subjects and objects, and arcs represent predicates. These relationships are illustrated in Figure 5. In the figure, the upper left node is a subject representing a resource. This subject is connected via predicates to the two objects labeled as object 1 and object 2. These two objects have literal values. Object 3 that the subject is connected to is also a resource. Thus, it is connected to another literal object labeled as object 4. For example, the subject that we are defining information about may be a URI representing a faculty in a university department. Object 1 and object 2 may be strings representing name and surname for that faculty. The faculty resource may be connected to an object representing a resource about a PhD student via the "advisorOf" predicate. Finally, the resource representing the PhD student may be connected to an object having a string value and representing "UniversityId" of the student.

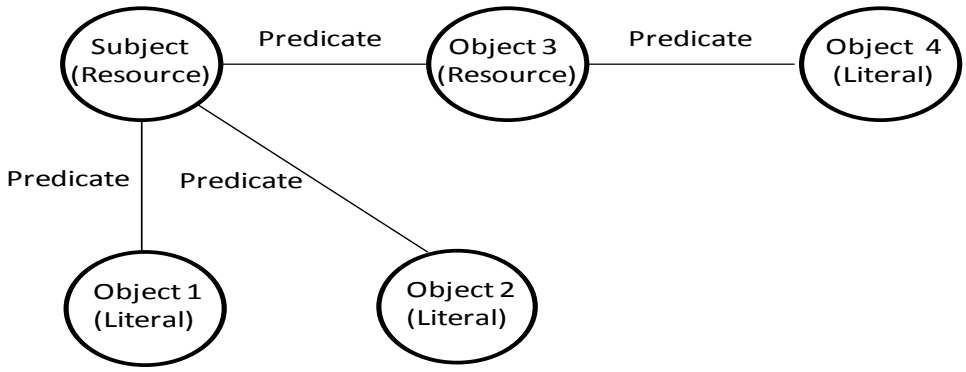

Figure 5. Resource Description Framework (RDF) statements on a graph data structure.

Linked data aims to transform a current web of documents into a global information space called "web of data". As a result of using RDF as the common standard for publishing and linking data, the "web of data" emerges as a big knowledge graph. Agents are the autonomous computing entities that will process the data in the linked data environment.

We think that gathering data from the linked data environment and internalizing that data as an agent's beliefs is one of the most important requirements that should be fulfilled in order to establish a connection between agents and linked data. Since data is gathered from the linked data environment by parsing the RDF documents, we need a graph based data model that will be used for representing the RDF triples internally in an agent.

Within the BDI architecture of LDAF-M, everything that an agent believes about its environment, including knowledge about other agents, is represented by the belief component. Any LDAF-M agent firstly obtains data from its environment. Subsequently, it associates a significant part of the obtained data with the belief component it contains. On the other hand, a LDAF-M agent can also obtain data from resources which have been developed in accordance with the principles of the linked data. Initially, a LDAF-M agent has been designed and implemented so that it can take its beliefs from Freebase [34]. Freebase provides documents and their relations expressed following the linked data principles. However, the belief component and architecture of LDAF-M has been designed and developed in such a flexible structure that data from other linked data sources such as DBpedia can also be obtained by easily integrating the necessary implementation to the system. LDAF-M includes an RDF parser for parsing these RDF documents and their relations. Since RDF documents form a knowledge graph, in LDAF-M a graph data structure that can represent data obtained in the form of RDF has been designed and implemented. The implementation model for the internal graph data structure is shown in Figure 6.

There are four components of the graph based data structure. These components are "GraphContainer", "GraphNode", "GraphPredicate" and "GraphProperty". A "GraphContainer" contains a certain number of "GraphNodes" (i.e., subjects or objects) and "GraphPredicates". The "GrapPredicate" concept establishes links between two nodes of a graph, hence represents the edges. Literals has the task of representing non-URI data related with a resource and is modeled using the "GraphProperty" class in Figure 6.

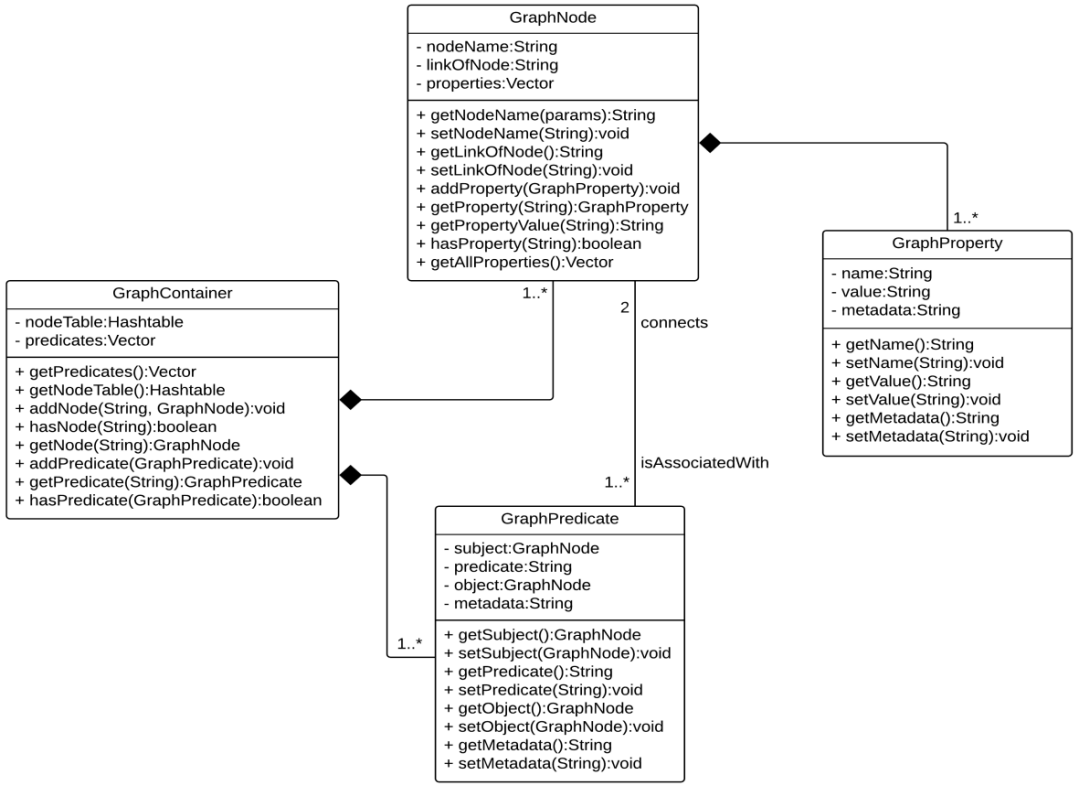

Figure 6. Implementation model for the internal graph data structure.

In the process of obtaining the data, first, the URI based links between RDF documents are discovered using a crawler-style mechanism. Then, each of the RDF documents discovered is parsed using the parser within the agent framework. Finally, a graph container is instantiated using the data obtained after parsing. Each "GraphContainer" represents a specific part of the beliefs stored in the belief base of an agent. Thus, an agent may contain more than one "GraphContainer" instance.

4. Case Study and Discussion

In the previous sections, we have given detailed information about the layers and structure of LDAF-M. In this section, we mainly introduce the agent based auction application that we have implemented as a case study.

Auctions are mechanisms by which buyers and sellers are in a continuous interaction and negotiate on scarce resources such as goods or utilities that they are going to sell or buy. As being autonomous computing entities, agents can carry out the bidding process on behalf of their users, can help in finding an optimal bidding strategy, and can collect information for their users during auctions. As a result, agents can facilitate many aspects in an online auction, and hence auction design and implementation is one of the well-known application areas of agents. We have chosen auction application, since there is a large number of interactions with many messages passing between agents in an auction scenario and this would be a good test environment for the communication infrastructure and planning mechanism of LDAF-M agents. The implemented scenario is an English auction.

We have structured this section in four parts. The first part introduces the main scenario of the agent based auction application that we have developed. The second part illustrates how to use the LDAF-M framework while building such an application. In particular, construction of a new agents, and definition of beliefs, goals and plans are explained. The third part gives a collaboration diagram that shows the interactions between agents and administrative components of the FIPA compliant LDAF-M agent platform as well as the interactions between agents themselves. Finally, a discussion constitutes the last part of this section.

4.1. Main Scenario

The auction application that we have implemented mainly allows users to exchange products among themselves. In this application, each one of the users may behave in two ways: They can either request to start an auction to sell a product or participate in an auction to buy a product.

A mobile device can play the role of a server which is responsible for managing an auction. In particular, it evaluates the requests coming from the clients for starting a new auction or participating in any of the ongoing auctions, and tracks the information flow in the auction. On the other hand, agents that run as clients on the mobile devices also play an important role for completing the deal. Information about the products and buyer and seller agents is represented using the belief component provided by the BDI model based agency layer of LDAF-M. Those beliefs that agents have about the products that are desired to be purchased or to be sold are obtained after processing Freebase data which is represented using the linked data principles.

4.2. Using the LDAF-M Framework

4.2.1. Constructing a New Agent

Using LDAF-M, developers are able to build agent based software which is mainly composed of agents running on mobile devices in various application domains.

When a developer needs to construct a new agent, he/she should extend the "Agent" class which serves as the main component for defining agents in LDAF-M. For example, the code fragment that is going to be used for building a client agent of the auction application is shown in Listing 1.

```
01  public class clientAgent extends Agent {
02      public void start() { ... }
03      public void finish() { ... }
04  }
```

Listing 1. Template for building a client agent.

The BDI related components of an agent, such as belief base, plan base, and goal base together with the other components, such as message base and agent life cycle related data, are held in the Agent class. The definitions of these components are made in the start() method. On the other hand, operations related with the termination of an agent are defined in the finish() method.

Agents constructed using LDAF-M framework communicate with each other using FIPA Agent Communication Language messages. An agent may receive many messages and to give the agent the ability to select the meaningful ones for itself among the incoming messages, a message filter class (e.g., FactoryMessageFilter) is defined in the framework. Using the methods in that class, developers can define patterns for indicating the significant messages that an agent expects. These pattern definitions are also made in the start() method of the agent class.

4.2.2. Defining Beliefs

As mentioned before, beliefs of an agent can be obtained dynamically from linked data resources as well as the other data sources defined by the developers. In this context, two different classes are used in defining the belief component of an agent. LinkedDataMetaBelief class which is extended from the MetaBelief class is responsible for representing beliefs coming from the linked data environment. It has a three-parameter constructor, using which a belief coming from a linked data resource is defined. The first parameter is the name given to the belief, while the second parameter is the URI of one of the related RDF documents that includes data that constitutes the content of the belief component. The third parameter is the name of the belief category that the defined belief belongs. The code fragment illustrating this process on an example belief in the auction application is given in Listing 2.

```
01 getBeliefBase().addGroupName("sell_product");
02 getBeliefBase().addGroupName("receive_product");
03 MetaBelief metaBelief = new LinkedDataMetaBelief("sell_product1",
04 "http://rdf.freebase.com/rdf/m.0pd3pb_", "sell_product");
05 getBeliefBase().addMetaBelief(metaBelief);
06 metaBelief = new LinkedDataMetaBelief("receive_product1",
07 "http://rdf.freebase.com/rdf/m.0pd3tfx", "receive_product");
08 getBeliefBase().addMetaBelief(metaBelief);
```

Listing 2. Defining linked data beliefs.

As shown in Listing 2, initially two belief categories, one for selling and the other for buying, are defined. Then, information about a product that is going to be sold is obtained from an RDF document in the Freebase environment and added as a belief to the agent's belief base. Similarly, another belief about buying a product is also added to the belief base of the agent.

The other class used in defining beliefs is the UserDefinedMetaBelief class which is extended from the MetaBelief class. This class is responsible for obtaining beliefs from the user defined data resources. It has a two-parameter constructor used in defining beliefs. The first parameter is the name given to the belief and the second parameter is the type of the data that constitutes the content of the belief component. The code fragment illustrating this process on an example belief in the auction application is given in Listing 3. The definition of two beliefs which are about an action and the participant list are given in Listing 3 respectively.

```
01 MetaBelief metaBelief = new UserDefinedMetaBelief("auction", Auction.class);
02 getBeliefBase().addMetaBelief(metaBelief);
03 metaBelief = new UserDefinedMetaBelief("participants", Vector.class);
04 getBeliefBase().addMetaBelief(metaBelief);
```

Listing 3. Defining other beliefs.

Both kinds of beliefs defined above are linked to the belief base of the agent by calling the addMetaBelief method that adds a belief passed to it as a parameter to the belief base of the agent.

4.2.3. Defining Goals

MetaGoal class plays the key role in defining an agent's goals. As you can see from the code fragment given in Listing 4, the MetaGoal class has a two-parameter constructor. The first parameter is the name of the goal, and the second parameter specifies the type of the goal. Some of the goals of an agent can be adopted either from the very beginning when the agent is activated or just before the agent terminates. These goals can be added to the goalbase of the agent using the methods addInitialGoalName and addFinishingGoalName as shown in Listing 4.

```
01  MetaGoal metagoal = new MetaGoal("finishing", MetaGoal.PERFORMGOAL);
02  getGoalbase().addMetaGoals(metagoal);
03  metagoal = new MetaGoal("initialization", MetaGoal.PERFORMGOAL);
04  getGoalbase().addMetaGoals(metagoal);
05  getGoalbase().addInitialGoalName("initialization");
06  getGoalbase().addFinishingGoalNames("finishing");
```

Listing 4. Adding goals to the goalbase of the agent.

As mentioned in Section 2.3, the life cycle of a goal can be controlled by the developer. The necessary conditions that are required for the transitions between the phases of the goal life cycle are defined in a class extended from the TriggeredCondition class. Such a class defined within the context of the developed auction application is shown in Listing 5. The AuctioneerAgentSuspendCondition class defines the conditions for an auctioneer agent to make a transition to suspended state and these conditions are written in the body of the method triggeredCondition().

```
01  public class AuctioneerAgentSuspendCondition extends TriggeredCondition{
02      public Boolean triggeredCondition(){
03          .....
04      }
05  }
```

Listing 5. Defining a condition for making a transition between two states.

The metaGoal class provides the methods for relating the defined conditions with the goal at hand. For example, setSuspendConditionClassName relates a suspend condition definition to a goal. On the other hand, the conditions that define when an agent has to drop a goal, is related with the goal via the method setDropConditionClassName. Examples for defining conditions for a goal that is related with starting an auction are shown in Listing 6.

```
01  metagoal = new MetaGoal("auction_process", MetaGoal.PERFORMGOAL);
02  metagoal.setSuspendConditionClassName
03  ("tr.edu.ege.bilmuh.application.sample.serverside.AuctioneerAgentSuspendCondition");
04  metagoal.setDropConditionClassName
05  ("tr.edu.ege.bilmuh.application.sample.serverside.AuctioneerAgentDropCondition");
06  getGoalbase().addMetaGoals(metagoal);
```

Listing 6. Defining conditions for a goal.

Another property in LDAF-M, which is a result of its BDI architecture, is that a goal can block some or all of the other goals during deliberation. For example, some of the goals of the agent can be

transferred to the option phase from the active phase, based on the blocking list of another goal that is being adopted by the agent. These blocked goals can never pass to the active state unless the blocking goal continues its execution. This information is provided through the addInhibitGoalName method located in the MetaGoal class. In addition, the number of maximum goals the agent can handle while executing a specific goal is defined using the setCardinality method as shown in Listing 7. In Listing 7, one can see that searching for an auction while sending an auction generation request is inhibited.

```
01 MetaGoal metagoal = new MetaGoal("send_request_generate_auction",
02 MetaGoal.PERFORMGOAL);
03 metagoal.addInhibitGoalName("search_auction");
04 metagoal.setCardinality(5);
05 getGoalbase().addMetaGoals(metagoal);
```

Listing 7. Setting properties for a goal.

4.2.4. Defining Plans

The MetaPlan class is the main class that supports the definition of plans which play an active role in transforming the goals into a series of actions. As illustrated in the code fragment given in Listing 8, MetaPlan class has a constructor with two parameters. The first parameter is the name of the plan being defined, and the second parameter is the name of the class containing the body of the plan. The class containing the body of a plan is an extension of the PlanContent class. In addition, it is also important to define the relationships between plans and goals, and the MetaPlan class includes a method using which such relationships can be defined. In Listing 8, the goal called "auction_process" is associated with the auction processing of the auctioneer agent which is responsible for managing the auction. Finally, the plan defined is added to the plan base of the agent.

```
01 MetaPlan metaplan = new MetaPlan("auction_process_plan",
02 "tr.edu.ege.bilmuh.application.sample.serverside.AuctioneerAuctionProcessPlan");
03 metaplan.addProbablyGoalName("auction_process");
04 getPlanbase().addMetaPlan(metaplan);
```

Listing 8. Defining plans.

In LDAF-M, execution of a plan not only starts with the adoption of a goal which associated with that plan, but also starts based on the occurrence of an event within the agent platform. For example, when an agent receives an agent communication language message from another agent, an event is created. Then, the plan associated with that event begins to execute provided that the relationship between that plan and the message generated the event have already been defined. This relationship is defined using the method addMessageEventTrigger which is located in the Planbase class. The use of that method is shown in Listing 9. The first parameter of the method is the reference name of the message trigger, and the second parameter is the name of the plan.

```
01 getPlanbase().addMessageEventTrigger("subscribe_auction_msg", "subscribe_auction");
02 getPlanbase().addMessageEventTrigger("get_auction_name_msg", "get_auction_name");
```

Listing 9. Adding event trigger to a plan.

The PlanContent class has also an important role in the execution of the plans, since the detailed operation of a plan is defined within that class. A plan content example is given in Listing 10.

The actions to be done within the scope of the plan are defined in the method called "body". Execution of a plan can end with success or failure. The actions to be done in case of success, just before the plan is terminated, are defined in the method called "passed", while the actions to be done in case of a failure are defined in the method called "failed". On the other hand, some plans cannot be

completed due to some reasons. When this is the case, the actions that should be done are defined in the method called "aborted".

```
01  public class EvaluateAuctionOfferPlan extends PlanContent {
02      public void body() { ..... }
03      public void passed() { ..... }
04      public void failed() { ..... }
05      public void aborted() { ..... }
06  }
```

Listing 10. Plan content.

From the agent developer's perspective, the plan content plays an important role, since a developer is able to define the following tasks and access the main components of an agent within a plan body:

- Interacting with the Agent Management System (AMS) to create an agent, to kill an agent, or to query the AMS about an agent.
- Accessing belief base, goal base, and plan base of the agent that is executing the plan body.
- Registering the services of an agent with the Directory Facilitator (DF), searching the DF for discovering the agents giving a specific service.
- Sending agent communication language messages to other agents.

4.3. Collaboration Diagram for the Case Study

The auction application which has been developed as a case study using the LDAF-M platform, is in fact a multi-agent system consisting of a number of agents. Each of these agents interacts with each other as well as with the LDAF-M platform's main components. We illustrate these interactions in a collaboration diagram which is given in Figure 7.

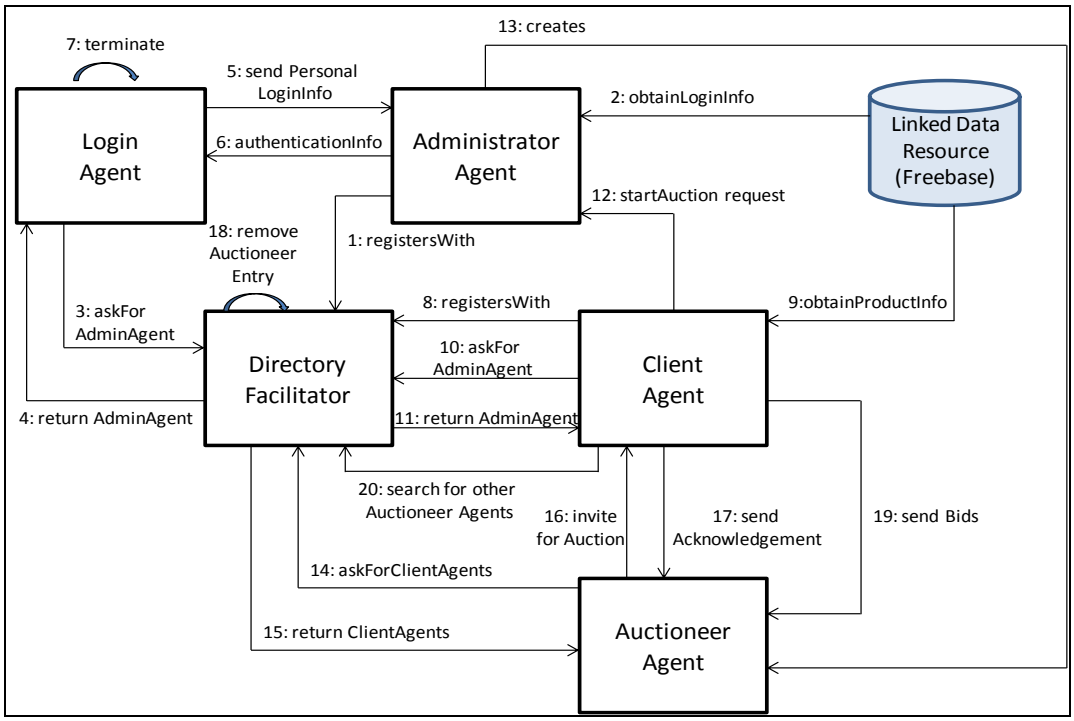

Figure 7. Collaboration diagram for the agent based auction application.

Each numbered interaction in the collaboration diagram is explained below. Please note that initially the agent platform is initialized, the AMS which will keep the agent identifiers and manage agent life cycle and the DF which is like a yellow page service are created. These steps are not shown for the sake of simplicity.

1: After the auction application has begun to execute on the mobile device that would serve as the server within the context of the developed application, an administrator agent is created. Upon creation, the administrator agent registers with the Directory Facilitator (DF) so that agents that are going to be created in future within the context of the auction application can discover it. The DF information console is shown in Figure 8.

Figure 8. The Directory Facilitator (DF) information console.

2: The administrator agent obtains the login information belonging to all of the system's users from a data resource. In our case, the administrator agent obtains this knowledge from the RDF documents that we have already recorded to Freebase and then internalizes that data in its belief base using the graph data structure introduced in Section 3.

3. On the client side, users that want to use the auction system need to create a login agent. Upon creation, the login agent asks the DF for the administrator agent.

4: The DF sends the information about the administrator agent to the login agent.

5: The login agent transmits its user's login information to the administrator agent.

6: If the authentication process ends with success, the user receives a login success message.

7: The login agent terminates after successful login.

8: After successful login, client agents which represent participants in an auction are created. Then they register with the DF.

9: A client agent obtains the data about the products that it is going to sell or buy from the RDF documents residing in Freebase. After the client agent parses that data, it maps the parsed data into the LDAF-M's graph based data structure to record them as its beliefs.

10: The client agent asks the DF for the administrator agent of the application.

11: The DF sends the information about the administrator agent to the client agent.

12: The client agent communicates directly with the administrator agent to send it a request for starting an auction for a product that it decides to sell. The client agent repeats sending this request for each of the products that it wants to sell.

13: After processing this request, the administrator agent creates an auctioneer agent which is responsible for managing the auction for that specific product.

14: The auctioneer agent asks the DF for the client agents registered.

15: The DF sends information about all the client agents which registered with it.

16: The auctioneer agent sends invitation messages to all of the client agents (other than the client agent which is the seller) asking them whether they prefer to participate in the auction for that specific product or not.

17: A client agent evaluates this invitation considering the beliefs in its belief base and makes a decision as a result. If it decides to participate in that auction, it notifies the auctioneer agent about this decision.

18: Each auctioneer agent registers itself with the DF for only a pre-specified time period. When the time period for an auctioneer agent completes, its registration record is removed from the DF. Upon this removal operation, the auction starts and participant client agents get a notification as shown in the screen given in the left part of Figure 9. The alert includes information about the product that is subject to the auction and the name and address of the client agent who plays the seller role.

Figure 9. Alerts indicating the start and end of an auction for a specific product.

19: Each participating client agent submits a bid to the auctioneer agent in sequence until the time period for the auction completes. The client agent who submitted the highest price bid wins the auction. Each participating client agent receives an auction completed alert including the seller, winner, and the price as shown in the screen given at the right part of Figure 9.

20: A client agent can also want to know about auction proposals that are suggested before that client agent has been created. When this is the case, the client agent communicates with the DF to discover auctioneer agents that are related to the products that the client agent wants to buy. After discovering the related auctioneer agents, the client agent sends a request to the auctioneer agent to indicate its decision in participating with the ongoing auction.

4.4. Discussion

There is no standard benchmark for evaluating the components and services provided by the LDAF-M, which is a FIPA-compliant and linked data aware agent development framework targeting mobile devices. Thus, we need to define a basic evaluation framework which defines different perspectives for the implemented framework. Those perspectives are listed below:

Perspective 1: We need to verify that the system fulfills the basic requirements given in the sub list below:

Requirement 1.1: The case when an agent(s) located in a mobile device becomes inaccessible should be detected and handled.

Requirement 1.2: Compliancy with the FIPA Agent Management Reference Model should be ensured.

Requirement 1.3: Agents supporting the BDI model can be developed using LDAF-M.

Requirement 1.4: Belief component within the BDI architecture should be able to obtain its beliefs also from the linked data environment and use them in the deliberation process.

Perspective 2: We need to observe the physical limits of the system in terms of the number of agents created together with the agent platforms.

Agent based auction domain has been selected, on purpose, as the domain of the application implemented as the case study, since this application includes several types of agents located on different platforms and constitutes a multiagent system where member agents intensively interact with each other as well as with the managerial components of the platform.

Several client agents playing the role of buyers in an auction and auctioneer agents playing the role of sellers of a specific product have been created. Some of the auctioneer agents have been intentionally made offline. Then, a client agent that has been created before those auctioneer agents is selected. When this client agent queries the DF to discover the auctioneer agents created before itself so that it can send a request to each of them for participating in ongoing auctions, it has been verified that the offline auctioneer agents are removed from the system and their related auctions are cancelled. This observation fulfills requirement 1.1.

The developed application also tests the FIPA-compliancy related components of the framework. Agent platforms have been constructed on several mobile devices and the services such as AMS and DF has been used. For example, life cycles of agents in the auction application are managed through AMS services. On the other hand, DF has been intensively used in the auction application. FIPA-ACL agent communication language messages have been sent during the interactions of agents. As a result, requirement 1.2 is fulfilled.

Within the scope of the developed auction application; beliefs, goals, plans and actions are defined for different kinds of agents such as buyer, seller, administrator and log-in agents. Those agents are executed successfully on their platforms and it has been observed that the deliberation mechanism inside each agent works. This fulfills the requirement 1.3.

To fulfill requirement 1.4, Freebase has been used as the linked data resource. Login information for the system users, as well as the data about products that are going to be exchanged, is supplied from the RDF documents located in Freebase. It has been observed that this data can be internalized as beliefs of the agents using the graph data structure defined inside the agents for this purpose.

The auction application has been implemented by installing the LDAF-M to several computers. One of them functioning as the server is the platform where the index server and the DF of the system reside. The index server plays a major role in managing the P2P communication infrastructure. The other ones, on which the LDAF-M has been installed to, correspond to mobile device agent platforms that include their own AMS services. Two auctioneer agents and four client agents are created and distributed over those platforms. Java multi-threading, which is used especially in the agency layer to support the BDI based deliberation process, is the primary factor affecting the physical limits of the system. The source code for agent based auction application is provided at the GitHub page of the project [29].

5. Related Work

In the literature, there are widely-known open source mobile-device agent frameworks which are JADE-LEAP, AFME, and 3APL-M.

JADE-LEAP (Java Agent Development Environment-Lightweight Extensible Agent Framework) [11] allows developing agent systems on mobile devices. In fact, LEAP is an add-on for the JADE framework. FIPA-compliant agent communication and execution infrastructure provided is its prominent feature. The main container in the JADE-LEAP framework includes both the DF and the AMS. On the other hand, in LDAF-M the index server agent platform, which is the equivalent component, includes only the DF. Instead, each mobile device platform includes its own AMS to prevent a possible overloading in the index server agent platform. In addition, the BDI model of agency is not supported in JADE-LEAP; it provides its own task execution infrastructure based on the different behavior types defined

AFME (Agent Factory Micro Edition) [13] is an agent framework which provides software developers the ability to develop agent-based applications in mobile platforms. AFME is built on top of the Agent Factory framework developed by the same research group. AFME supports the BDI model of agency and represents beliefs as logical sentences in its belief base.

3APL-M [12], is another mobile device agent development framework that supports the BDI model of agency. It is built upon the Artificial Autonomous Agents Programming Language (3APL) language from which it takes the deliberation mechanism. The beliefs of agents in 3APL-M are represented using Prolog language.

However, data that represent the beliefs of the agents running within LDAF-M platform can be obtained from the data sources in the linked data environment. This gathered data is internalized using the graph data structure defined in LDAF-M for this purpose. This support for linked data is the main difference of LDAF-M.

On the other hand, after the introduction of Android operating system, several studies have been conducted to develop agents for Android based mobile devices. While some of these studies aimed to extend previous agent development frameworks for supporting Android based devices, the other studies aimed to develop agent platforms for Android based devices from scratch. A new JADE configuration that addresses Android running mobile devices [15] is an example for the first category. Additionally, a new JADEX version that supports implementing agents on Android based mobile devices has been released [16].

Andromeda (ANDROid eMbeddED Agent) platform [35] is an example for the second category. The aim of the Andromeda project is to provide a development platform to support users in developing embedded agents for mobile devices using the Android operating system. It provides the basic building blocks such as Agent, Behavior, Task, and Capability. These components are integrated with the Android system's building blocks. On the other hand, Andromeda supports neither FIPA standards nor BDI model of agency.

JaCa-Android [36] is another platform for developing mobile applications. As its developers also emphasize, JaCa-Android is not a new agent development platform and does not aim to port existing agent technologies into mobile devices. It rather presents a programming model for developing Android based smart mobile applications using the Jason agent development framework [9] and the CArtAgO environment programming framework [37].

When this study was initiated, we decided to begin the development of LDAF-M using Java technology for mobile devices (e.g., Java Platform, Micro Edition–Java ME) [38]. Java ME is supported by many different kinds of mobile devices and mobile device operating systems by means of the different Java ME profiles. On the other hand, undoubtedly the Android operating system has found widespread use in recent years. Based on this fact, there is a tendency towards developing an Android configuration for mobile device agent frameworks that have originally been developed using Java technology, as seen in the case of Android configurations for JADE and JADEX. There are several bridges to convert Java ME applications into Android applications, but we foresee that we might spent serious manual effort for porting LDAF-M which is rather a comprehensive Java ME application. Being focused on the linked data support in the current version of LDAF-M, but accepting the fact that Android operating system occupies an important place in the mobile applications ecosystem, porting the presented framework into the Android environment is left as the primary future work.

A comparison table that compares LDAF-M and other frameworks from four perspectives is given in Table 1 below. Please note that JaCa-Android is not included in the comparison table, since it is not a new agent framework intended to be developed specifically for mobile devices.

Table 1. Comparison table for LDAF-M and other mobile device agent frameworks.

Framework Name	P2P Communication Infrastructure	FIPA Compliancy	BDI Model of Agency	Linked Data Support
JADE LEAP	YES	YES	NO	NO
3 APL-M	NO	NO	YES	NO
AFME	YES	YES	YES	NO
JADEX-Android	NO	YES	YES	NO
ANDROMEDA	NO	NO	NO	NO
LDAF-M	YES	YES	YES	YES

An important research area in the multi-agent systems field is agent based simulation. There are many techniques proposed and tools developed in this area [39]. One such technique proposed for developing agent based simulation applications is TABSAOND [40]. TABSAOND has a software framework that allows the deployment of agent based simulation models as mobile applications. This is an important characteristic for a simulation tool, since users' access to desktop computers can be limited in domains such as crisis management. The software framework of TABSAOND has been developed using the JavaScript programming language. This is also an important property, since it is possible to generate cross-platform native mobile applications from JavaScript codes. LDAF-M does not support the development of agent based simulations, it is rather a general agent development framework that can either be used in developing multi-agent systems in any domain that requires cooperative or competitive behavior, such as information retrieval, and e-commerce as long as the appropriate plans are defined, or be used in developing intelligent personal digital assistants that are in continuous interaction with the other components and data resources in the system. On the other hand, we will be inspired by TABSAOND, when we think of porting the presented framework to JavaScript environment to take the advantages of cross-platform development as a future work.

Another perspective that we want to cover in the related work section is the coordination of agents. Coordination in multi-agent systems is a closely related topic with communication, since a coordination mechanism is built on top of a communication infrastructure. In a nutshell, in LDAF-M, agents use the communication infrastructure for exchanging agent communication language messages with other agents. In addition, life cycle management of an agent (i.e., detecting whether an agent is alive or off-line) is handled through this infrastructure. Since an agent can behave autonomously on behalf of its users, and can interchangeably play the roles of a client and a server, it is possible to describe an agent as a "peer". The communication infrastructure employs the appropriate network programming and server-side programming technologies to give agents (peers) the ability to communicate with each other, and to support agent life cycle management. In LDAF-M, coordination can be realized through the agent plans that are manually defined by the developers for a specific application.

In the literature, there are several coordination patterns defined for multi-agent systems. One such coordination pattern is the one described by Magariño and Gutiérrez, considering a crisis management scenario where there are collaborative agents interacting with both each other and the environment [41]. The implementation of this pattern includes coordination, network and information agents. Coordination agents live in the mobile devices and are responsible for coordinating people in the area in case of a crisis. Network agents behave as an intermediary in communication of the agents. Finally, information agents keep knowledge of locations related with the crisis and the city map. In LDAF-M, the peers can communicate directly with each other, there is no intermediary service. The index server in LDAF-M platform is responsible for maintaining the life cycle of agents, it does not behave as an intermediary during communications.

On the other hand, coordination patterns can be included in LDAF-M as pre-built interaction protocols. In this way, developers can choose the appropriate protocol during design. Support for built-in interaction protocols has already been provided by the JADE framework. Contract net and brokering are examples for JADE framework's pre-built interaction protocols. The same approach can also be employed in LDAF-M.

Studies that only aim to use semantic web/linked data technologies from mobile applications form the final perspective regarding the related work, though these studies do not consider agent technology. For example, DBpedia mobile is a location aware application that uses location data obtained from DBpedia [42]. A case study in the tourism domain has been developed to demonstrate the DBpedia mobile system. Although this study is a valuable effort that aims to use semantic web technologies in mobile devices, it does not address agent technology.

6. Conclusions

In this study, the FIPA-compliant agent framework which has been developed for mobile devices is introduced. Providing P2P communication infrastructure, following FIPA Agent Management Reference Model specification, and having an agency layer that implements the BDI model of agency are prominent features of this framework. On the other hand, an agent which is developed using the provided framework is able to construct its belief base by internalizing the data obtained from the resources in the linked data environment. Linked data support is the distinguishing feature of the developed framework that makes it different from other mobile device agent frameworks.

An agent based auction application has been developed as the case study using the framework. This application having a number of different agents, requiring many interactions between agents themselves and the administrative components of the platform, and making agents to obtain their beliefs from Freebase as the linked data resource, has sufficiently provided the context to test the framework. On the other hand, the implementation is flexible enough to add new linked data sources such as DBpedia.

As mentioned in the introduction section, when recent studies are inspected, one can see that use of agent technology is still in progress in popular domains such as cyber physical systems and internet of things among others. We think that agent development frameworks, which specifically address mobile devices, will be of great importance for being used in developing agent based applications in such domains.

Author Contributions: Conceptualization, İ.S.B. and R.C.E.; Software, İ.S.B.; Supervision, R.C.E.; Validation, İ.S.B.; Writing—original draft, İ.S.B. and R.C.E.; Writing—review and editing, R.C.E.

Funding: This research received no external funding.

Acknowledgments: This work has been conducted during the Master degree thesis study of İlker Semih Boztepe under the supervision of Rıza Cenk Erdur at Ege University Computer Engineering Department. During the Master degree education period, İlker Semih Boztepe has been awarded a scholarship by the Scientific and Technological Research Council of Turkey (TUBITAK).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wooldridge, M. *An Introduction to Multiagent Systems*, 2nd ed.; John Wiley & Sons Ltd.: Chichester, UK, 2009; pp. 22–47, ISBN 9780470519462.
2. Wooldridge, M.; Jennings, N.R. Intelligent agents: Theory and practice. *Knowl. Eng. Rev.* **1995**, *10*, 115–152. [CrossRef]
3. Wooldridge, M. *An Introduction to Multiagent Systems*, 2nd ed.; John Wiley & Sons Ltd.: Chichester, UK, 2009; pp. 201–219, ISBN 9780470519462.
4. Parunak, H.V.D. A practitioners' review of industrial agent applications. *Autonom. Agents Multi-Agent Syst.* **2000**, *3*, 389–407. [CrossRef]

5. Leitao, P.; Karnouskos, S.; Ribeiro, L.; Lee, J.; Strasser, T.; Colombo, A.W. Smart agents in industrial cyber–physical systems. *Proc. IEEE* **2016**, *104*, 1086–1101. [CrossRef]
6. Carlier, F.; Renault, V. IoT-a, Embedded Agents for Smart Internet of Things. Application on a Display Wall. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), Omaha, NE, USA, 13–16 October 2016; IEEE Computer Society: Los Alamitos, CA, USA, 2016; pp. 80–83. [CrossRef]
7. Bellifemine, F.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; John Wiley & Sons Ltd.: Chichester, UK, 2007; pp. 1–286, ISBN 978-0470057476.
8. Pokahr, A.; Braubach, L.; Lamersdorf, W. Jadex: A BDI reasoning engine. In *Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations*; Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A., Eds.; International Book Series; Springer: Boston, MA, USA, 2005; Volume 15, pp. 149–174, ISBN 978-0387245683. [CrossRef]
9. Bordini, R.H.; Hübner, J.F.; Wooldridge, M. *Programming Multi-Agent Systems in AgentSpeak Using Jason*; John Wiley & Sons Ltd.: Chichester, UK, 2007; pp. 1–273, ISBN 978-0470029008.
10. Kravari, K.; Bassiliades, N. A survey of agent platforms. *J. Artif. Soc. Soc. Simul.* **2015**, *18*, 1–18. [CrossRef]
11. Leap User Guide. Available online: http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf (accessed on 19 August 2018).
12. Koch, F.; Meyer, J.J.C.; Dignum, F.; Rahwan, I. Programming Deliberative Agents for Mobile Services: The 3APL-M Platform. In Proceedings of the Programming Multi-Agent Systems (ProMAS 2005), Utrecht, The Netherlands, 26 July 2005; Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 222–235. [CrossRef]
13. 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 the International Conference on Computational Science (ICCS2006), Reading, UK, 28–31 May 2006; Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 727–734. [CrossRef]
14. Rao, A.S.; Georgeff, M.P. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, USA, 12–14 June 1995; Lesser, V., Gasser, L., Eds.; MIT Press: Cambridge, MA, USA, 1995.
15. Bergenti, F.; Caire, G.; Gotta, D. Agents on the Move: JADE for Android Devices. In Proceedings of the XV Workshop From Objects to Agents ("Dagli Oggetti agli Agenti"), Catania, Italy, 26 September 2014; Volume 1260.
16. Jadex Android. Available online: https://download.actoron.com/docs/releases/jadex-3.0.69/jadex-mkdocs/android/android/ (accessed on 25 September 2018).
17. Berners-Lee, T.; Hendler, J.; Lassila, O. The Semantic Web. *Sci. Am.* **2001**, *284*, 34–43. [CrossRef]
18. Wood, D.; Zaidman, M.; Ruth, L.; Hausenblas, M. *Linked Data: Structured Data on the Web*, 1st ed.; Manning Publications Co.: Shelter Island, NY, USA, 2014; pp. 1–276, ISBN 978-1617290398.
19. Hausenblas, M. Exploiting linked data to build web applications. *IEEE Internet Comput.* **2009**, *13*, 68–73. [CrossRef]
20. FIPA—The Foundation for Intelligent Physical Agents. Available online: http://www.fipa.org/ (accessed on 19 August 2018).
21. Rao, A.S.; Georgeff, M.P.; Kinny, D. A Methodology and Modelling Technique for Systems of BDI Agents. In Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW' 96), Eindhoven, The Netherlands, 22–25 January 1996; Van de Velde, W., Perram, J.W., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 56–71. [CrossRef]
22. Wooldridge, M. *An Introduction to Multiagent Systems*, 2nd ed.; John Wiley & Sons Ltd.: Chichester, UK, 2009; pp. 65–84, ISBN 978-0470519462.
23. Bratman, M.E. *Intention, Plans, and Practical Reason*, 1st ed.; Harvard University Press: Cambridge, MA, USA, 1987; pp. 1–200, ISBN 9780674458185.
24. Abiona, O.O.; Oluwaranti, A.I.; Anjali, T.; Onime, C.E.; Popoola, E.O.; Aderounmu, G.A.; Oluwatope, A.O.; Kehinde, L.O. Architectural Model for Wireless Peer-to-Peer (WP2P) File Sharing for Ubiquitous Mobile Devices. In Proceedings of the IEEE International Conference on Electro/Information Technology (EIT 2009), Windsor, ON, Canada, 7–9 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 35–39. [CrossRef]

25. Vu, Q.H.; Lupu, M.; Ooi, B.C. *Peer-to-Peer Computing—Principles and Applications*, 1st ed.; Springer: New York, NY, USA, 2010; pp. 1–317, ISBN 9783642035142. [CrossRef]
26. Steinzment, R.; Wehrle, K. *Peer-to-Peer Systems and Applications*, 1st ed.; Springer-Verlag: Berlin/Heidelberg, Germany, 2005; pp. 1–615, ISBN 9783540320470. [CrossRef]
27. FIPA Agent Management Specification. Available online: http://www.fipa.org/specs/fipa00023/ (accessed on 19 August 2018).
28. Wooldridge, M. Intelligent agents. In *A Modern Approach to Distributed Artificial Intelligence*, 1st ed.; Weiss, G., Ed.; MIT Press: Cambridge, MA, USA, 2000; p. 58, ISBN 9780262731317.
29. GitHub Page for LDAF-M. Available online: https://github.com/ilkersemih/SemanticWebAgent (accessed on 1 October 2018).
30. Berners-Lee, T.; Linked Data. World Wide Web Design Issues. Available online: https://www.w3.org/DesignIssues/LinkedData.html (accessed on 19 August 2018).
31. Bizer, C.; Heath, T.; Berners-Lee, T. Linked data–the story so far. *Int. J. Semant. Web Inf. Syst.* **2009**, *5*, 1–22. [CrossRef]
32. Liyang, Y.A. *Developer's Guide to the Semantic Web*, 2nd ed.; Springer-Verlag: Berlin/Heidelberg, Germany, 2014; pp. 1–829, ISBN 9783662437964.
33. SPARQL Query Language for RDF. Available online: http://www.w3.org/TR/rdf-sparql-query/ (accessed on 19 August 2018).
34. Freebase. Available online: https://developers.google.com/freebase/ (accessed on 19 August 2018).
35. Agüero, J.; Rebollo, M.; Carrascosa, C.; Julian, V. Developing intelligent agents in the Android platform. In Proceedings of the 6th European Workshop on Multi-Agent Systems, Bath, UK, 18–19 December 2008.
36. Santi, A.; Guidi, M.; Ricci, A. Jaca-Android: An Agent-based Platform for Building Smart Mobile Applications. In Proceedings of the Languages, Methodologies, and Development Tools for Multi-Agent Systems (LADS 2010), Lyon, France, 30 August–1 September 2010; Dastani, M., El Fallah Seghrouchni, A., Hübner, J., Leite, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 95–114. [CrossRef]
37. Ricci, A.; Piunti, M.; Viroli, M. Environment programming in multi-agent systems: An artifact-based perspective. *Autonom. Agents Multi-Agent Syst.* **2011**, *23*, 158–192. [CrossRef]
38. Java Platform, Micro Edition (Java ME). Available online: https://www.oracle.com/technetwork/java/embedded/javame/index.html (accessed on 27 September 2018).
39. Abar, S.; Theodoropoulos, G.K.; Lemarinier, P.; O'Hare, G.M.P. Agent based modeling and simulation tools: A review of the state-of-art software. *Comput. Sci. Rev.* **2017**, *24*, 13–33. [CrossRef]
40. Magariño, I.G.; Palacios-Navarro, G.; Lacuesta, R. TABSAOND: A technique for developing agent-based simulation apps and online tools with nondeterministic decisions. *Simul. Model. Pract. Theory* **2017**, *77*, 84–107. [CrossRef]
41. García-Magariño, I.; Gutiérrez, C. Agent-oriented modeling and development of a system for crisis management. *Expert Syst. Appl.* **2013**, *40*, 6580–6592. [CrossRef]
42. Becker, C.; Bizer, C. DBpedia Mobile: A Location-enabled Linked Data Browser. In Proceedings of the WWW2008 Workshop on Linked Data on the Web (LDOW 2008), Beijing, China, 22 April 2008; Bizer, C., Heath, T., Idehen, K., Berners-Lee, T., Eds.; Volume 369.