Pintea *et al. Complex Adapt Syst Model* (2018) 6:10
https://doi.org/10.1186/s40294-018-0061-7

Complex Adaptive Systems Modeling

RESEARCH Open Access

Multi-agents features on Android platforms

Camelia-M. Pintea[1*], Andreea Camelia Tripon[2], Anca Avram[1] and Gloria-Cerasela Crişan[3]

*Correspondence:
dr.camelia.pintea@ieee.org
[1] Technical University Cluj-Napoca, 28 Memorandumului, Cluj-Napoca, Romania
Full list of author information is available at the end of the article

Abstract

The current paper shows the multi-agents capabilities for valid and flexible applications when using a framework. Agent-based functions were used within JADE framework for an Android messaging application with all requirements included. In the paper are described the architecture, the main functions and the databases integration of a user friendly agent-based application. There are included existing and possible multi-agents characteristics to provide integration with mobile platforms and storage challenges to improve the user experience through data mining.

Keywords: Meta-heuristics, Intelligent agents, Mobile platforms

Mathematics subject classification: MSC I.2.8, MSC I.2.11

Introduction

Multi-agent systems are involved today for solving different types of problems. They could be used in real-time applications and for solving complex problems in different domains as bio-informatics, ambient intelligence, semantic web (Jennings et al. 1998; Warneke et al. 2001; Wooldridge 2013). The main properties of agents are the autonomy, reactivity, pro-activeness, cooperation, mobility and not at last learning capabilities. With the exception of the last one, all of them are implemented in the presented JADE platform for Android. The autonomy refers to the ability of each client to perform a connection and to enjoy the service from the platform. The reactivity of an agent is provided by the graphic interface, which listens and reflects the events taking place on the server. The pro-activeness feature is implemented as sensing the automatic Smart Lock function of Android. Cooperation is offered by the group management functions, and mobility is offered by the intrinsic characteristics of both JADE and Android.

JADE (Java Agent Development Framework) is a Java-based framework used to specify the multi-agent systems. A JADE-based system can be distributed over many systems and its configuration could be controlled for example by a user-friendly graphical application. As is designed as a development framework for heterogeneous, distributed agents, JADE lacks mechanisms for intelligent planning or reasoning. If such features are sought, then the JADE-based application needs interactions connections with Prolog or JESS modules.

The communication architecture of JADE offers an efficient and flexible transmission of messages based on a private queue of messages under *Agent Communication*

Language (ACL) format for each agent. The agents can easily identify the ACL messages received from other agents and have access to their queue of messages (Finin et al. 1997).

The Android applications are nowadays some of the most used application, especially due to the open operating system features. Known to be frequently used, the messaging applications are real-time communication means between people. The messaging package could be installed and used through any existing Android devices as smartphones or tablets.

The combination between the broad set of features of ACL, JADEs interaction protocols and the ubiquitous presence of Android brings an important value to the development of distributed and decentralized applications. The goal of this paper is to propose such a combination, in order to provide an extensible and multi-platform messaging application. There are other approaches for developing such Android applications. For example, the packages presented in Multi-User chat (2018), MIT project MUC (2018) use XMPP protocol for multi-user text chat (XMPP protocol 2018), similar to IRC.

The paper briefly presents the multi-agents with specific features for Android applications in the second section. In "A new messaging application based on JADE" section the messaging application based on JADE is introduced, with the description of the structure and functionalities. "Theoretical approaches on new agent features for Android applications" section includes new approaches on agent features for Android applications including GPS/GIS characteristics. "Storage challenges and improving the user experience through data mining" section illustrates the storage challenges and how to improve the user experience through data mining. The conclusion of the paper presents the new JADE-based messaging and some multi-agent future improvements.

Multi-agent systems enhance the sndroid applications features

Intelligent agents have been used in several disciplines as Artificial Intelligence, in human-computer interface design and in object-based systems (Jennings et al. 1998).

The agents are inspired from real living “agents” (Camazine et al. 2001; Michener 2003) as humans, insects or other animals capable to react to their own environment, with own objectives to reach and the autonomous capability to make the proper activities to achieve their goals (Chira et al. 2009; Pintea et al. 2012).

In general, it is assumed that an agent has the following properties (Wooldridge 2013):

- The autonomy shows the ability of the agent to operate by itself without any other intervention.
- The reactivity shows the ability of the agent to know the environment and react to the changes from its environment.
- The pro-activeness shows the ability of the agent to have initiative and pursue its own goals.
- The cooperation shows the capability of the agent to interact with others, agents or humans, through a specific communication language.
- The learning ability of the agent is activated while the agent interacts with its environment.
- The mobility shows the ability of an agent to move in a self-directed way around a network.

Based on the particularities of a problem to be solved, the agents could be endowed with other features as for example rationality or sensitivity. The agents from the *Multi-Agent System (MAS)* are autonomous and heterogeneous agents capable of interaction. In MAS the computation is asynchronous and has no global control (Jennings et al. 1998; Russell and Norvig 2009).

In Jennings et al. (1998) is specified that negotiation, as a coordination process, is essential in *MAS* to solve conflicts. The communication Jennings et al. (1998) in *MAS* is a must due to exchanging information or other inter-operation tasks between agents. The communication process between agents requires both the *Agent Communication Language (ACL)* and understanding the concepts exchanged by agents.

The specific literature Poslad (2007), Android Developer (2018); Kikuchi et al. (2017) presents many connections between the evolution and the organization of a MAS and the way an Android application works and is coded. These strong connections generated in fact the JADE project, its worldwide success both in academia and in industry.

Basically, the event-driven paradigm materialized by a generic Android application can be seen as the theoretical specification of the behavior of MAS. In the following it is presented such an application that couples the MAS paradigm with Android.

Previous applications for Android have been published. For example (Chat demo for Android 2018) is a basic, official demo from JADE. A chat room Client based on JADE is presented at Chat room client under JADE (2018). In Caire et al. (2012) it is presented a hybrid application with a non-specific GUI (Fig. 1). These models could be extended with multiple user features. This application is such an extended messaging package.

A new messaging application based on JADE

An application based on JADE is a set of components called agents. The agents are uniquely identified by their names. The agents execute tasks and interact by exchanging messages. The agents are kept "alive" through a platform offering services. A platform has one or many containers that could be run on systems with different hosts. Each container could have zero or more agents. Each platform has a special container called *Main Container* (Fig. 2).

The minimal installation and running requirements are: minimum JDK 5, Android Studio and Android SDK. Once Main Container is started, the server starts too. The

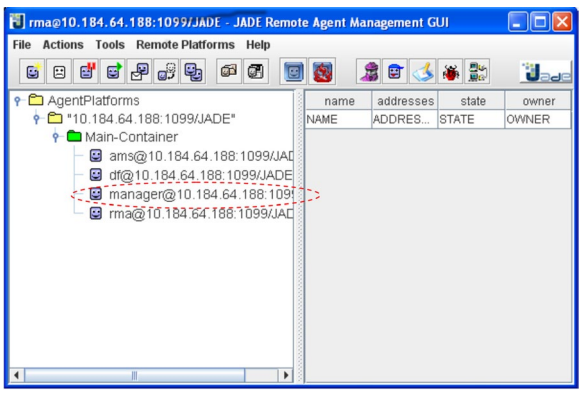

Fig. 1 The main container with the chat manager agent (Caire et al. 2012)

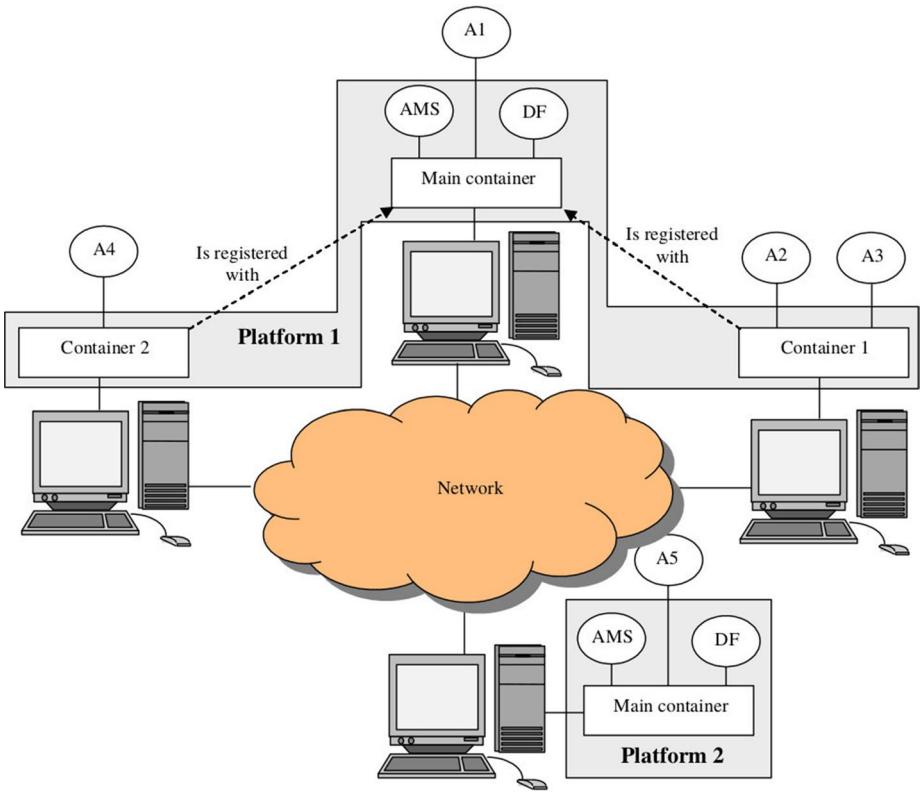

Fig. 2 Containers and platforms in JADE framework (JADE Framework 2018)

other agents will run from the same or different system and will connect with Main Container. After that, the application is ready to function as a real-time messaging application.

When an agent is connected to Main Container, a new container will be created and will be visible; to run other agents we can install the messaging application on an Android phone, run the application and then connect to Main Container (Fig. 3).

Main Container is the container that starts at the beginning and the other containers will login to; Main Container includes two special agents:

- The *Agent Management System (AMS)* is the authority of the platform and it is the only agent that could manage the platform (starting or stopping agents or stopping the entire platform).
- The *Direction Facilities* (DF) is a service through which the agents could publish the tasks offered and give the possibility to find other agents based on the task.

The communication between agents is made no matter if the agents are or not in the same container or if they belong or not to the same platform. The messages format ACL is defined by *Foundation for Intelligent Physical Agents (FIPA)* FIPA (2018). An ACL message includes the sender, the receiver, the communication task and the content of the message. There are twenty two communication FIPA sets, each one well defined and having a well defined semantic. For example INFORM message:

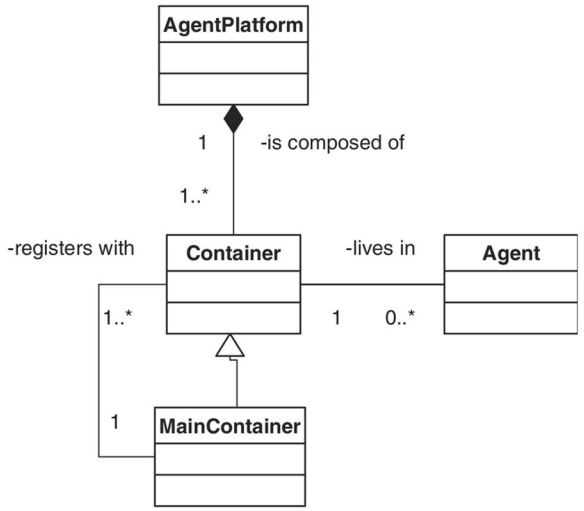

Fig. 3 Relationship between the main architectural elements of a JADE application (Bellifemine et al. 2007)

"INFORM 7 May 2018 rains", or REQUEST message requesting the receiver to make a task.

Applications using LEAP have several restrictions (Caire and Pieri 2011). They could not replicate the Main Container, so the security issue has to be treated with supplementary resources. The AMUSE project (AMUSE project 2016) reports in its current version only direct terminal-to-terminal conversations, with no support for graphics and GUI development.

The messaging application

A device (phone/tablet) with Android operating system connects with the server IP address; after the connection is established, the agent could send and receive messages.

The application considers the client-server principle: through the already existing connections, the clients ask the server to do some tasks by sending messages. When the server receives the message, the server sends it further to all the connected clients. The application has two modules:

1. The *Android messaging Client* permits the messaging connection from an Android device; it includes a graphical interface and an agent managing the interactions with the other components.
2. The *messaging Server* is the platform called Main Container including an agent called *ChatManagerAgent* which control all agents connected to the messaging application.

The steps to connect to Main Container are the following:

– The installation of application on the Android phone or on a simulator.
– The IP address of the server is set from the menu and should be the same with the physical device where Main container is.

– The user must choose a name, which should be different from the name of the other messaging-users.
– If there are no errors, the messaging-user could send messages to the other messaging-users.

The structure of the application

The application has two parts. The first part makes the connection with the JADE agent and the second one is the graphical interface. They are included in the packages `agent` and `gui`, respectively. Based on the Android architecture, the JADE agent is integrated in the project through *jade-Android.jar* library; it includes the following services to connect to the server: *RuntimeService* and *MicroRuntimeService*. In this particular application the MicroRuntimeService is used. The connection with the peers is made by the class MainActivity through the following code included in the startChat method:

```
serviceConnection = new ServiceConnection(){
  public void onServiceConnected(ComponentName name, IBinder service){
    microRuntimeServiceBinder = (MicroRuntimeServiceBinder) service;

  };
  public void onServiceDisconnected(ComponentName name){
    microRuntimeServiceBinder = null;
  }
};

bindService(new Intent(getApplicationContext(),
 MicroRuntimeService.class),serviceConnection,
 Context.BIND_AUTO_CREATE);
```

Once connected to the service, we can go to the next step, which is to create the container and to start the agent. If it is a success, then the application can send and receive messages. As a JADE-messaging novelty, there are included many actions further described including for example the groups of users, the messages with a single user, the messages within a group, blocking/unblocking users, etc.

```
void getAllUsers();
void getUsersNotInGroup(Group group);
void getBlockedUsers();
void getGroupsConversations();
void getUsersFromGroup(Group group);
void getMessagesFromUser(User user);
void getMessagesFromGroup(Group group);
void getUserConversations();
void createGroup(Group group);
void leaveGroup(Group group);
void addToGroup(User user, Group group);
void blockUser(User user);
void unblockUser(User user);
void register(User user);
void login(User user);
void sendMessageToGroup(Message message);
void sendMessageToUser(Message message);
void deleteMessage(Message message);
ArrayList<User> getParticipantNames();
```

The first method is called when a message is sent from the phone and the second when the user wants to see the other messaging-users connected to the same server.

These two actions are on the `ChatClientAgent` class, implementing the `ChatClientInterface`. This class deals with sending and receiving the messages further to/from the `ChatActivity` class in order to visualize the messages. In the `ChatClientAgent` class it is implemented a listener; when the server send a message, the message is shown through a broadcast receiver.

The database design is presented in Fig. 4. This model has five tables: Users, with the registered users of the application, Blocked for storing the pairs (Blocking, Blocked), Groups for the registered groups, Users_groups for the members of each group, and Messages for the content and side characteristics of each message.

The main functions of the application

The introduced application is a stand-alone messaging application for Android with a user-friendly design. The application starts from the main menu, after the user logged in or registered to the application. If some of the settings are not correct, the user could change the port or the host address. To set the connection the address where the JADE agent would be connected, the host and the port should be specified. Implicitly the previous ones are used.

Logging into the system is beneficent to keep all the user information saved and the user could use different mobile phone to send/receive the messages, or see the previous messages.

The button used to show all the users will open a page with the status of all registered users. The current user could add any user of an existing group or block another user. If a user is blocked, he does not have the possibility to send messages to the one who blocked him. In the same context, the blocker cannot see the status of an blocked user and cannot add him to a group. These could be possible after the user is unblocked (Fig. 5).

A new group is created easily using a particular button and by specifying its name. The name is verified to be unique in the database and an appropriate message is shown. The new group is automatically added to the other existing groups. Nevertheless, the only ones who can add other members to a group are the members of the group.

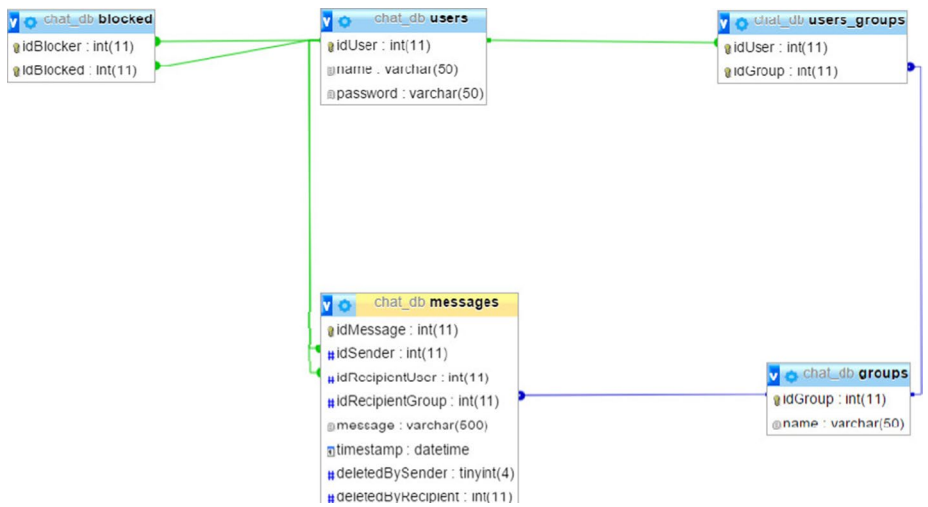

Fig. 4 Database tables and relationships for the proposed application

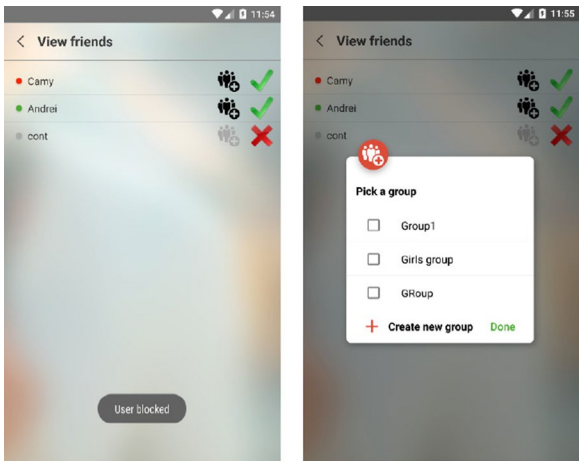

Fig. 5 Examples of the messaging-JADE features: blocking/unblocking users (left) and managing groups of users (right)

When selecting a conversation with a friend all the sent and received messages from that user will be shown. On the toolbar there are the user-name, a button to add the friend to the group, a button to add the user in the blocking list and a button to delete all the conversation. In the left side of the window there are the received messages and in the right side the sent messages. All the messages include the time and date when they were sent. At the bottom of the window are the buttons for adding and sending new messages. To delete a message, one can use a long touch on that message and the message could be deleted after a confirmation.

For the conversations in a group there is a similar functionality; the differences are on the toolbar: adding a new member to the group, visualization of all the group members and their status, the button to leave the group. All the messages received from any member of the group will be placed in the left side of the window and the sent ones in the right side of the window.

Theoretical approaches on new agent features for Android applications

Several common features for Multi Agent Systems and Android applications are:

- Concurrency: the Android components are activated by intents (decentralized events trigger an agent behavior or the Yellow Pages service provided by the DF special JADE agent (JADE Framework 2018)).

  The agents from MAS act simultaneously, in a common environment, pursuing their individual goals, but possibly ready to draw coalitions or to co-operate. For example, the method sendMessageToGroup is designed to produce a broadcast in the group context.
- Loosed coupling: like agents in MASs, the Android application target components are activated by explicit or implicit intents (Poslad 2007). The method register acts as remote command on the server.

- Asynchronous communication: both for the agents in MAs and for the Android application components, the self-decided conversation initiation is a manifestation of the individual, autonomous behavior. The method getGroupsConversations is executed at client connection, so it provides delays in communication.
- Flexibility: Android is a multi-channel, multi-carrier, freely distributed OS, based on Java, and has a market share of 88% in 2018 Smartphone market share (2018). The entire platform is designed to be hardware-independent.
  The implicit intent object is an example of how the Android operating system adapts to the environment (Android Developer 2018). Likewise, current complex social problems are modeled by MASs that need to perform well in open environments.
  For example, the MIT Robust Open Multi-Agent Systems (ROMA) Research Group is dedicated to "learning how we can develop multi-agent systems for open contexts where the constituent agents can come from anywhere, may be buggy or even malicious, and must run in the dynamic and potentially failure-prone environments at hand".

The agents involved in the JADE messaging application have limited features. Here we propose several new features to be included in the JADE framework.

The Multi-agent system of JADE could include agents with different levels of sensitivity related to their environment. Based on the agents sensitivity, the messaging application could include other facilities as:

- The messaging could start when the user is closer to a already set point; for example one could set the messaging application to start when approaching to a building, let's say a museum, when the application starts automatically based on the GPS/GIS data; so, the agent is sensitive to each "museum" encountered to exchange impressions with the other users.
- The messaging could include a blocking option based on the agent sensitivity: an user is automatically blocked or unblocked based on his behavior.
  - If the words used by another user are "bad", so they are in a database, locally or cloud-based, the user is automatically blocked and an appropriate message is sent to the other user;
  - An user could be unblocked if it is identified a good behavior on its personal social platform; for example on its social website volunteering is included, or "good words" and facts are promoted. This feature could be added or not by the user.
- Another improvement could be considered the automatically connection to a close or extended group when an earthquake is ongoing based on a specific sensor of the device or by an earthquake alert. Similar features could be included for floods or other related crisis.
  The GPS/GIS features could be used as in Moratis et al. (2004), Cai (2005), Lei and Hui (2006), Crișan et al. (2016), Android GIS project (2018). In Cai (2005) for example is shown an implementation of an intelligent, multi-modal, multi-user

geographic information environment (GCCM_Connect) used in a spatial decision-making contexts. Collaboration and shared knowledge are needed especially when is involved a critical problem as for example earthquakes or floods. Crisis management demands information technology and individuals /organizations to share information and expertise. Apps based on GIS for iOS and Android are developed by Esri Explorer for ArcGIS (2018).

– An idea of improvement in **terms of functionality** would be to have the existing list of name servers available and users to choose from the server list instead of having to configure the server himself. When blocking another user, one could invoke a reason for doing that if he/she wants. There could be a list of predefined reasons or user can invoke a new one. Later on, analysis of data could show the potential users that have malicious intents. Auto-complete features could be provided based on analyzing the most used phrases and offer edit support to the users using the application.

All the new and the common characteristics generate tight connections, allowing the researchers to design and to provide useful integration of the mobile platform with Multi Agent System features.

Storage challenges and improving the user experience through data mining

For the messaging application presented, there was the need of having information stored in a database. The information that is stored consists of details related to users, login information, friends of a user, blocked/unblocked users; messaging groups; messages exchanged between users; agent behavior.

As soon as the application is used by more and more users, the volume of the data increases and a proper database maintenance plan needs to be in place. That means there will be a need for performing actions like:

– Cleaning unnecessary data-logs of the status of actions performed by the agents for example, are only needed for a short amount of time (in case of an error for example, these logs could provide a meaningful support for finding the cause of the problem).
– Archiving the old data. A proper system will make sure that data that is not in use anymore is handled (for example messages exchanged in groups that no longer contain active users). Archiving could mean storing the plain data to a different server (less performing, less expensive). Another approach would be obtaining a more compact version of that data and storing it in a different format.
– Having an indexing strategy that is reviewed periodically. Indexes are data structures that improve the speed of data retrieval operations on a table. Having proper indexes will mean a faster time to obtain the results, but reviewing the indexes periodically is a must, because the increase of data could mean that an index that was performing at some point might no longer be efficient.

The database could also provide the means for obtaining relevant information related to the behavior of the application and agents by performing data mining. Currently data mining is widely recognized as the process of discovering relevant patterns in large sets of data, patterns that can be later used. To have these patterns discovered, intelligent

methods are applied. Data mining can be applied to any kind of data, as long as it delivers meaningful results to a target application (Han et al. 2011; Russell and Klassen 2018).

In the context of the JADE messaging application, data mining could be applied on the stored data related to the users to obtain relevant information on the user behavior. For example, discovering the use patterns for a specific functionality could give an idea on what type of users are using that functionality, what are the topics that are the most tackled, what is the geographical distribution of the users, whether the application is more widely used in the urban or rural area. This type of knowledge could lead further on to developing new functions that are targeted to improving user experience; remove/make less visible functions that do not come with an adequate return of investment.

Furthermore, in Cao et al. (2009) the concept of agent mining refers to the application of autonomous intelligent agents in the field of data mining to support and improve the knowledge discovery and decision-making process. Because agents have autonomy, flexibility, mobility, adaptability and have a rational nature, they prove to be a perfect choice for parallel, multisource, distributed mining. In the context of the JADE framework, that could mean integrating agents responsible with mining the available data.

Conclusions and future work

The paper illustres multi-agents features on applications using specific frameworks. JADE framework is illustrated while successfully developers use components for a messaging application as the *Agent Communication Language (ACL)* to send/receive the messages. The user-friendly application includes several features as using groups of users or blocking/unblocking users on any Android devices. Further improvenments focusing on data mining and specific multi-agents features, as sensitivity based on GPS/GIS location are shown.

Authors' contributions

Overall, the authors equally contribute equally to the manuscript. Specifically, CMP conceived the idea of the paper. All the authors developed the simulation models. ACT executed the simulation experiments. All authors wrote the paper. All authors read and approved the final manuscript.

Author details

[1] Technical University Cluj-Napoca, 28 Memorandumului, Cluj-Napoca, Romania. [2] RodeApps, 3–5 Campul painii, Cluj-Napoca, Romania. [3] "Vasile Alecsandri" University, 157 Calea Mărăşeşti, Bacău, Romania.

Competing interests

The authors declare that they have no competing interests.

Funding

The authors received no specific funding for the manuscript.

Publisher's Note

Received: 23 August 2018 Accepted: 9 November 2018
Published online: 16 November 2018

References

AMUSE project (2016) http://jade.tilab.com/amuseproject/
Android Developer (2018) https://developer.android.com/guide/components/intents-filters.html
Android GIS project (2018) http://www.opengis.ch/android-gis/
Bellifemine FL, Caire G, Greenwood D (2007) Developing multi-agent systems with JADE, vol 7. Wiley, Hoboken
Cai Guoray (2005) Extending distributed GIS to support geo-collaborative crisis management. Geogr Inf Sci 11(1):4–14. https://doi.org/10.1080/10824000509480595
Caire G, Iavaronne G, Izzo M, Heffner K (2012) JADE programming for Android http://jade.tilab.com/doc/tutorials/JadeAndroid-Programming-Tutorial.pdf

Caire G, Pieri F (2011) LEAP User Guide http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf
Camazine S et al. (2001) Self organization in biological systems. Princeton University Press, Princeton. http://press.princeton.edu/titles/7104.html
Cao C, Gorodetsky V, Mitkas PM (2009) Agent mining: the synergy of agents and data mining https://doi.org/10.1007/978-3-642-15420-1
Chat demo for Android (2018) http://jade.tilab.com/documentation/chat-demo/
Chat room client under JADE (2018) https://github.com/satish2/JADE-Android-App
Chira C, Pintea CM, Crisan GC, Dumitrescu D (2009) Solving the linear ordering problem using ant models. In: GECCO. ACM, New York, 2009, pp. 1803–1804 https://doi.org/10.1145/1569901.1570170
Crişan GC, Pintea C-M, Palade V (2016) Emergency management using geographic information systems: application to the first Romanian Traveling Salesman problem instance. Knowl Inf Syst 50(1):265–285. https://doi.org/10.1007/s10115-016-0938-8
Explorer for ArcGIS (2018) https://www.esri.com/en-us/arcgis/products/explorer-for-arcgis
FIPA (2018) http://www.fipa.org/
Finin T, Labrou Y, Mayfield J (1997) KQML as an agent communication language software agents. Jeffrey BM (ed.) https://doi.org/10.1145/191246.191322
Han J, Kamber M, Pei J (2011) Data Mining: Concepts and Techniques, The Morgan Kaufmann Series. https://doi.org/10.1016/C2009-0-61819-5
JADE Framework (2018) http://jade.tilab.com
Jennings NR, Sycara KP, Wooldridge M (1998) A Roadmap of agent research and development. J Auton Agents Multi-Agent Syst 1(1):7–36. https://doi.org/10.1023/A:1010090405266
Kikuchi K, Cetinkaya A, Hayakawa T, Ishii H (2017) Stochastic communication protocols for multi-agent consensus under jamming attacks 2017. In: IEEE 56th annual conference on decision and control (CDC), Melbourne, VIC 2017, pp 1657–1662. https://doi.org/10.1109/CDC.2017.8263888
Lei Ye, Hui Lin (2006) The Web integration of the GPS+GPRS+GIS tracking system and real-time monitoring system based on MAS web and wireless geographical information systems: W2GIS Hong Kong 2006, pp 54–65 https://doi.org/10.1007/11935148_6
MIT Robust Open Multi-Agent Systems. http://ccs.mit.edu/roma/
MIT project MUC (2018) http://web.mit.edu/svalente/lib/smack_3_0_4/documentation/extensions/muc.html
Michener CD (2003) The social behavior of bees: a comparative study. Ann Rev Entomol 14(1):299–342. https://doi.org/10.1146/annurev.en.14.010169.001503
Moratis P, Petraki E, Spanoudakis NI (2004) Providing advanced, personalised infomobility services using agent technology. In: Applications and innovations in intelligent systems. Springer, London. XI pp 35–48 https://doi.org/10.1007/978-1-4471-0643-2_3
Multi-User chat (2018) https://xmpp.org/extensions/xep-0045.html
Pintea CM, Crisan GC, Chira C (2012) Hybrid ant models with a transition policy for solving a complex problem. Logic J IGPL 20(3):560–569. https://doi.org/10.1093/jigpal/jzr004
Poslad S (2007) Specifying protocols for multi-agent systems interaction. ACM Trans Auton Adapt Syst 2(4):15. https://doi.org/10.1145/1293731.1293735
Russell M, Klassen M (2018) Mining the social web: data mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and more, 3rd edn. OReilly Media, Sebastopol
Russell S, Norvig P (2009) Artificial Intelligence: a modern approach, 3rd edn
Smartphone market share (2018) http://www.idc.com/prodserv/smartphone-os-market-share.jsp
Warneke B, Last M, Liebowitz B, Pister KSJ (2001) Smart Dust: communicating with a cubic-millimeter computer. Computer 34:44–51. https://doi.org/10.1109/2.895117
Wooldridge M (2013) Intelligent agents an introduction to multiagent systems, In: Weiss G (ed) MIT Press, Cambridge
XMPP protocol (2018) https://xmpp.org/about/