SilentMeet - A Prototype Mobile Application for Real-Time Automated Group-Based Collaboration

Remus-Alexandru Dobrican, Gilles I. F. Neyens and Denis Zampunieris
University of Luxembourg
Luxembourg, Grand-Duchy of Luxembourg
Email: remus.dobrican@uni.lu, gilles.neyens.001@student.uni.lu, denis.zampunieris@uni.lu

***Abstract*—Today's growing world of mobile devices offers all the necessary elements for developing collaborative mobile applications. However, this brings new challenges like how to handle the high complexity of efficient collaborative mechanisms or automatize part of the user's interaction with the applications, as too many actions are required from the users in order to perform even the most basic operations. This paper describes an experimental mobile application, i.e., SilentMeet, that uses a rule-based middleware architecture for mobile devices and a new technique for exchanging information, for coordinating and for taking distributed decisions. More precisely, the application is designed to detect, based on collaboration, possible meetings or events with more than 2 participants and automatically switch the smartphone into silent mode. The goal of SilentMeet can be divided into 2 two main parts: 1) to develop a collaborative application with the help of rule-based systems; and 2) implement and evaluate Global Proactive Scenarios (GPaSs) in a real-case example.**

Keywords–mobile devices; collaborative applications; proactive computing; distributed group collaboration.

I. INTRODUCTION

Communication and collaboration, more precisely interactive collaboration, are two key aspects in today's mobile world. Basic mobile applications that are able to perform only local tasks do not address the increasing needs of the users anymore. The demand for services and applications which support communication and collaboration of mobile devices has raised significantly in the past years [1]. The latest interest in mobile collaboration can be explained by the large number of mobile devices around the world, which is continuing to grow from one year to another [2]. However, this mobile environment capable of performing distributed operations brings new challenges, such as intermittent connectivity, data heterogeneity, limited computational capabilities and users' mobility. Also important, is the fact that mobile networks, due to the high mobility of their users [3], differ a lot from static systems, where the users are always connected. This leads to the issues like determining the context information needed to trigger the collaboration process or like users being temporarily unavailable while they are still engaged in the collaborative operations.

Another important aspect to be addressed, when designing collaborative applications, is to establish up to which level will the users interact with the system. Because users may have basic skills or only limited experience when interacting with complex applications or because they do not want to spend a lot of their time giving instructions to the system, the applications can automatize a lot of their processes. One of the solutions for doing this is Proactive Systems, which are able to act on their own initiative and to take decisions on behalf of their users [4]. Recently, the possibility of implementing a Proactive Engine for mobile devices was investigated [5]. The added value is that, with the help of a mobile Proactive System, which is essentially an advanced rule-based system, developers can directly add the functionality they want to their applications by using Proactive Rules. From the developer's point of view, a Proactive Rule represents a tool for writing a set of instructions, while from the system's point of view, a Proactive Rule is a piece of code which has to be executed. More about Proactive Rules and examples with the rules used for this study will be shown in Section III.

In order to have a rule-based system capable of executing Proactive Rules on mobile devices, a middleware model was created for Android-based devices [6]. This represents an important achievement as until now only lightweight basic rule-based engines like [7] and [8] were developed for mobile platforms. These engines would allow applications to use simple conditional rules. The middleware model is also providing an information sharing method between the devices called Global Proactive Scenario (GPaS) [9]. This method was implemented to give the possibility to the applications to perform collaborative tasks.

Numerous studies [3][10][11] have been conducted that provide middleware architectures as tools for developing collaborative applications. One important difference is that these studies look at collaboration from a different angle. More precisely, they concentrate on user-centred collaboration, where the focus is to get the users to interact more and more with their applications on the mobile devices. The issue is that these applications would depend too much on the actions of their users and, if the users do not engage properly in each step of their interaction with their devices, the applications may remain at the same step. Opposite to this, Proactive Computing, which was defined by Tennenhouse as a new way of computing, for and on behalf of the user [12], tries to reduce the users' involvement by automatizing some processes. By doing so, the users can concentrate more on the most important parts of the collaboration.

Many mobile applications exist on the market, like Silence[13], Go Silent [14] or Advanced Silent Mode [15], which automatically switch off the sounds of mobile devices based on the user's preferences. These simple applications perform only local tasks like checking the user's predefined preferences or detecting calendar events. They do not use

 ISBN: 978-1-61208-436-7

any kind of collaboration with other devices to make the application smarter.

For example, SilentTime [16] searches for weekly events in the local schedule and automatically silences the user's phone if a future event is detected. It offers the user the possibility to add exceptions, in case he/she is waiting for an important phone call. However, the application has a couple of downsides. First, it is exclusively based on the users input, i.e., a calendar event or an exceptions of a special situations will only be detected if the user creates them before, and second, it does not use any kind of communication with other devices to check if the events will take place or not. Another example is AutoSilent [17], which is slightly different from SilentTime because it adds an extra step of verification before muting the users phone, i.e., it will verify if the users location corresponds with the events location at a certain time. This extra feature is again just a simple check because it does not use any kind of collaboration, like, for example, checking also the location of the other participants.

In this study, we investigate how a mobile application, i.e., SilentMeet, which uses a proactive rule-based middleware system to communicate and collaborate, is automatically turning the devices into silent mode if a meeting is detected and confirmed between a predefined group of users.

The rest of the paper is structured as follows. Section II introduces the problem statement and a motivating scenario that points out the need for automatizing certain tasks and processes inside applications in order to reduce the user's involvement in unnecessary situations. Section III contains explanations about SilentMeet's architecture and about its way of reaching a global decision based on distributed reasoning, including how and which Proactive Rules were used in this case. Tests on real devices are discussed in Section IV and their results in Section IV-B. And finally, Section V contains the main conclusions and future work.

II. Motivating Scenario

There are quite a few mobile users who went through embarrassing situations when their phones rang during important meetings, lectures, exams, presentations, concerts, interviews or key talks offered at international conferences. Imagine, for example, that during a viola recital of a famous musician, the mobile phone of a person start ringing, like it did during a recital in Slovakia [18]. The musician is not only interrupted but he/she could also loose focus and find it difficult to continue. The are many more other examples when muting the phone is a mandatory requirement. The main problem is that each user has to manually configure his/her phone to be silent during important events. And often, they forget. A general common strategy or approach which performs collaborative actions is missing.

Let's imagine the following scenario: an important event is about to begin. The mobile devices of the participants, located in their pockets, go automatically into silent mode. The participants do not have to worry they forgot to silence their mobile phones, they can focus more on their important tasks. The meeting can continue without any interruptions or embarrasing situations.

III. A rule-based solution - SilentMeet

SilentMeet is a collaborative application which is developed in order to minimize the risk of interruptions and their distracting effects during an important event such as a meeting, interview or public event. Moreover, for having an efficient distributed algorithm, part of the user's actions are automatized with the help of Proactive Rules. We assume that groups of people are predefined when an event is created by each user. More precisely, when a calendar event is created, the user also adds the participants. Users can perform collaborative actions only if they are part of the same group of the same event. So, users first have to build their own groups or agree to be part of already created groups. For example, in a company, the secretary of a department creates a group for the employees of that department that have meetings regularly. By joining this group, the members agree that their mobiles phones can be silenced by the application of the other members, after multiple rounds of negotiation. More about the negotiation process is presented in Section III-D. Also, more conditions and checks are taken into account like the location of the event and the participants, the date and the hour of the event and the local preferences of each user.

A. Middleware model - Proactive Engine for Mobile Devices

The Proactive Engine is a middleware architecture developed to support the execution of Proactive Rules. It was designed to perform background operation and to interact with the user only when necessary. Moreover, it comes equipped with a Rules Engine to process rules, a data storage mechanism to store different parameters and with a communication layer, to be sure the Proactive Engine is able to share important information.

Proactive Engines communicate with each other by sending JavaScript Object Notation (JSON) messages. The messages can contain questions, answers or commands, depending on their purpose. For example, a Proactive Engine can send a question to another engine to ask for various context information. Based on the received answer, if some conditions are fulfilled, the engine can then send a command to the other engine to perform an action. Messages are forwarded to a server and to the cloud. The server and the cloud are in charge of assigning each device with a device ID and with forwarding the JSON messages to the targeted devices. They also handle special cases such as lost JSON messages or devices that are not temporary available on the network.

B. Global Proactive Scenarios (GPaSs)

The idea of SilentMeet is that the devices participating in a collaboration process can take global decision, not only local ones. Each device is able to make use of the global knowledge, which is created by all the devices. For example, a basic application would only be able to detect an event based on the local information provided by the calendar of a device. SilentMeet is able to query all the devices to obtain more precise information about that event with the help of GPaSs. A GPaS is a data exchange mechanism, which allows devices to dynamically acquire relevant context information by merging data from multiple sources. It works between all mobile devices with an integrated Proactive Engine. GPaSs,

 ISBN: 978-1-61208-436-7

from a technical point of view, are composed of sets of Proactive Rules.

SilentMeet contains one GPaS, as it uses a distributed reasoning algorithm to reach a decision and to execute specific tasks. In this specif GPaS, each device needs additional information from the other devices before taking a decision. The idea is that if multiple devices, part of a collaboration group, have an event in their local calendar, with the same date, time and location, it is very probable that the event will take place. We presume that the same information about an event coming from 2 different devices part of the same group is enough for the application to decide what to do next. In this case, it will switch the corresponding devices into silent mode when the event will take place. The minimum number of 2 devices is motivated by the fact that a device should not be able to mute, by itself, other devices without any kind of agreement. Also, a decision can be taken without the confirmation of the event from all the participants, as this is very difficult to achieve in real-life situations, where each user is expected to manually add the event into the calendar.

C. Proactive Rules

Proactive Rules, as shown in Figures 1, 2, 3, 4 or 5, contain a set of instructions, which are written by the developer. These rules are to be executed by the Proactive Engine when different events are detected or when they are missing. The initial structure of a Proactive Rule [19] was used for creating the rules necessary for SilentMeet. It contains 5 main parts such as *data acquisition*, *activation guards*, *conditions*, *actions* and *rules generation*. These parts are important as they decide when a rule is executed, if the rule performs its actions, if the rule will generate other rules or will just simply clone itself. Proactive Rules can have different execution times because their activation depends on the local settings of each device and on the user's actions. For example, 2 users creating a new calendar event at different hours on their phones, trigger, at different time intervals, the rule which starts the negotiation process of SilentMeet.

For achieving its goal, SilentMeet only needs 5 Proactive Rules. SilentMeet will come together with the 5 Proactive Rules, when installed on each device. Initially, only one Proactive Rules, the first one, will be executed by the Proactive Engine. Then, all the rules can be activated, if their execution conditions are meet.

Illustrated in Figure 1, the first Proactive Rule, i.e., R001-DetectMeeting, is used to detect new meetings added in the calendar of each device. Adding the meeting in the calendar was either manually added by the user or automatically added by another application. If a new meeting is detected, all the participants of that meeting will be retrieved and will be contacted by a second rule. If the first rule is not activated, it will continues to clone itself for being executed at the next iteration of the Proactive Engine. When activated, rule R002 - ContactAttendees, shown in Figure 2, is in charge of sending a request to all the participants for more information about the meeting. The request is then forwarded with specific parameters like the sender ID, the destination ID and the full details of the event itself.

R001 - DetectMeeting

Description: This Rule is the initial rule in the proactive engine. It will check for new events in the calendar and then create a ContactAttendees rule for each event.

```
parameters
    None
data acquisition
    Event[] events= getNewEvents();
activation guards
    return !empty(events);
conditions
    return true;
actions
    None
rules generation
    if (!activationGuard());
        cloneRule(DetectMeeting);
    else
        foreach event in events:
            createRuleContactAttendees(event);
    end if
```

Figure 1. First Proactive Rule in pseudo-code

R002 - ContactAttendees

Description: This Rule sends an AskMeetingConfirmation Rule to every attendee of an event.

```
parameters
    Event event;
data acquisition
    String[] attendees= event.getAttendees();
activation guards
    return true
conditions
    return true;
actions
    foreach attendee in attendees:
        sendAskMeetingConfirmationRule(attendee,event,deviceID);
rules generation
    None
```

Figure 2. Second Proactive Rule in pseudo-code

The third rule, or rule R003 - AskMeetingConfirmation, presented in Figure 3, sends back a response about the event to the device which has previously sent the request for extra information. The answer is positive if an event with the same date, hour, location and participants is detected in the calendar and a negative answer otherwise. The fourth rule, R004 - ConfirmMeeting, checks for the answers of each participant and, if there is at least one confirmation, it validates the requirements of having at least 2 users that will attend the same meeting. If the previous condition is meet, it activates the last rule, which is in charge of muting the mobile phones during a selected event. And the last rule, R005 - MuteCommand, is the one that checks if the meeting is about to begin, and, if the device's location is close to the location of the event, it will activate the silent mode that that particular device. The local preferences of a the user are also checked because there are situations where the user is expecting an important phone call. For example, a man would like to be called if his pregnant wife is giving birth at the hospital, even if, at the time of the call, the man would be in a meeting.

 ISBN: 978-1-61208-436-7

R003 - AskMeetingConfirmation

Description: This Rule will check whether an event is present in the calendar of this user. If this is the case it will send a ConfirmMeeting Rule back to the device which started the negotiation.

parameters
Event event;
String senderDeviceID;

data acquisition
boolean answer=meetingExists(event.startTime,event.endTime);

activation guards
return answer;

conditions
return true;

actions
sendConfirmMeetingRule(senderDeviceID,event);

rules generation
None

Figure 3. Third Proactive Rule in pseudo-code

R004 - ConfirmMeeting

Description: Upon receiving this Rule the application will issue a MuteCommand Rule to the device that confirmed the meeting.

parameters
Event event;

data acquisition
None

activation guards
return true;

conditions
return true;

actions
sendMuteCommandRule(senderDeviceID,event);

rules generation
None

Figure 4. Fourth Proactive Rule in pseudo-code

R005 - MuteCommand

Description: This Rule will silence the phone when the event starts.

parameters
Event event;

data acquisition
None

activation guards
return eventStarted(event);

conditions
return atMeetingLocation(getCurrentLocation(), event.getLocation()) and checkPreferences(event));

actions
mutePhone();

rules generation
if (!activationGuard()):
cloneRule(MuteCommand);
end if

Figure 5. Fifth Proactive Rule in pseudo-code

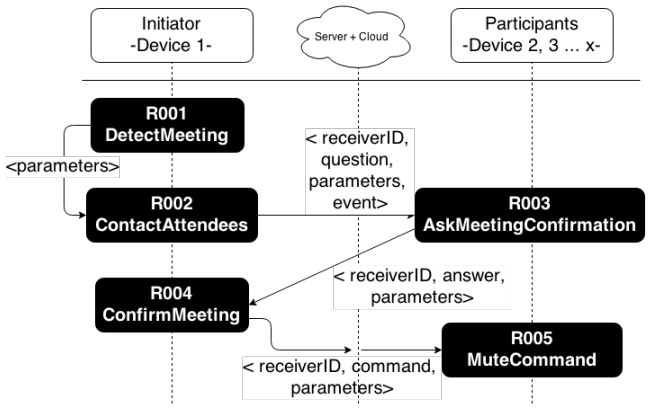

Figure 6. Sequence diagram with the rules activation and the communication process of SilentMeet

D. Negotiation Process

For muting the mobile devices of the participants of a group, after a calendar event is detected, SilentMeet passes through a couple of rounds of negotiation. As shown in Figure 6, when a device detects an event in its calendar, with the help of rule R001, it immediately checks for the participants of that event. Then, it starts, with the help of rule R002, contacting and asking each participants about the event. On the receiving devices, rule R003 is activated and start looking for an event in the calendar with the particular characteristics as the ones received in the list of parameters. An answer is then forwarded to the initiating device and rule R004 is activated. Rule R004 will check if the Initiator receives at least one positive answer, i.e., event detected on another device, it will send a command to the participants which confirmed the event to activate rule R005. The last rule will then be activated and it will wait for all the conditions to be satisfied in order to perform its actions, i.e., to put the device on silent mode.

IV. TESTS

Tests were conducted locally at our university on 3 different devices: a Samsung Galaxy Note 3, a Samsung Galaxy S3 and a Nexus 4, as shown in Figure 7. All 3 devices use an Android operating system and have the Proactive Engine middleware installed in order to be able to execute rules and collaborate with each other in GPaSs. The devices were part of a predefined group of 3 participants. During the tests, all 3 devices were connected via WiFi to the same network. Initially, all the devices had their sound turned on and had an event in their calendar with the same date, time and location. The first rule, R001 - DeetectMeeting, was activated on each device when the event started to take place. The Rule Engine was set to execute the rules present in its queue every 30 seconds.

The participating devices used in the tests were part of the same predifined group for the given event. For registering to the group, the user of each device had to use a unique email address, e.g., user1@uni.com, an email address provided by the user of the Galaxy Note 3.

 ISBN: 978-1-61208-436-7

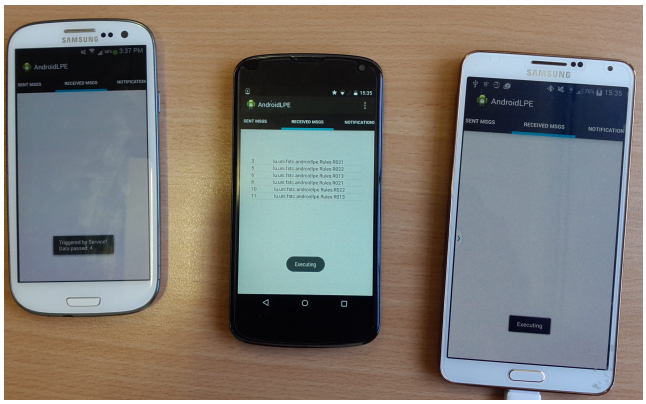

Figure 7. Devices used for testing SilentMeet, in the process of activating the silent mode

A. Measurements

The main goal was to check if the application behaved as expected and, based on the algorithm for distributed agreement, if all three devices were muted, after checking if there is at least one other person who is still attending the meeting. Another point of interest of the tests was the time the devices needed to reach a common agreement and to perform the actions of the last rule R005 - MuteCommand, which muted all the devices.

B. Results and discussions

The tests showed that the application behaves as expected and that all three devices were muted after the negotiation process. In the given settings, it took around 10 seconds to reach a common agreement that the meeting will take place and to mute all three devices. However, this time is highly dependent on the frequency parameter of the Rule Engine, meaning that setting a lower time interval between two iterations will also lead to a faster execution of the GPaS.

V. CONCLUSION AND FUTURE WORK

In this paper, we show that it is possible to develop a collaborative application on top of a rule-based middleware engine and with the help of Proactive Computing, more precisely by using Global Proactive Scenarios. The application is able to detect and acquire relevant context-information, use a distributed reasoning algorithm and take global decision. At the same time, several parts of the collaboration process were automatised and the user's involvement reduced only to the most important operations.

Future work includes developing more complex collaborative applications and other Global Proactive Scenarios on top of the Proactive middleware engine. One possible direction is to develop another GPaS for turning off the sounds of each smartphone, based on the location of the participants. If the location of the participants would correspond to the location of the event on the given date the Proactive Engines would start to alert each other and mute each smartphone. This would show how more than one GPaS can be used in the same application.

REFERENCES

[1] Forrester. Latest IT Trends For Secure Mobile Collaboration. Forrester Consulting. [Online]. Available: http://www.connectedfuturesmag.com/docs/byod_forrester_tap_latest_it_trends_wp_en.pdf [retrieved: May, 2015]

[2] CISCO. VNI Mobile Forecast Highlights. CISCO Systems. [Online]. Available: http://www.cisco.com/c/dam/assets/sol/sp/vni/forecast_highlights_mobile/index.html [retrieved: May, 2015]

[3] V. Sacramento and et al., "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users," Distributed Systems Online, IEEE, vol. 5, no. 10, Oct 2004, pp. 2–2.

[4] A. Salovaara and A. Oulasvirta, "Six modes of proactive resource management: a user-centric typology for proactive behaviors," in Proceedings of the third Nordic conference on Human-computer interaction. ACM, 2004, pp. 57–60.

[5] R.-A. Dobrican and D. Zampunieris, "Moving Towards Distributed Networks of Proactive, Self-Adaptive and Context-Aware Systems: a New Research Direction?" The International Journal on Advances in Networks and Services, vol. 7, 2014, pp. 262–272, ISSN: 1942-2644.

[6] G. I. F. Neyens, R.-A. Dobrican, and D. Zampunieris, "Enhancing Mobile Devices with Cooperative Proactive Computing," COLLA - The Fifth International Conference on Advanced Collaborative Networks, Systems and Applications, 2015, to be published.

[7] M. Slazynski, S. Bobek, and G. J. Nalepa, "Migration of Rule Inference Engine to Mobile Platform. Challenges and Case Study," in Proceedings of 10th Workshop on Knowledge Engineering and Software Engineering (KESE10) co-located with 21st European Conference on Artificial Intelligence (ECAI 2014), Prague, Czech Republic, August 19 2014., 2014. [Online]. Available: http://ceur-ws.org/Vol-1289/kese10-08_submission_4.pdf

[8] C. Choi, I. Park, S. J. Hyun, D. Lee, and D. H. Sim, "MiRE: A minimal rule engine for context-aware mobile devices," in Third IEEE International Conference on Digital Information Management (ICDIM), November 13-16, 2008, London, UK, Proceedings, 2008, pp. 172–177.

[9] R. Dobrican and D. Zampunieris, "A Proactive Approach for Information Sharing Strategies in an Environment of Multiple Connected Ubiquitous Devices," in Proceedings of the International Symposium on Ubiquitous Systems and Data Engineering (USDE 2014) in conjunction with 11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2014). IEEE, 2014, pp. 763–771.

[10] J. Gabler, R. Klauck, M. Pink, and H. Konig, "uBeeMe - A platform to enable mobile collaborative applications," in Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on, Oct 2013, pp. 188–196.

[11] P. Coutinho and T. Rodden, "The FUSE Platform: Supporting Ubiquitous Collaboration Within Diverse Mobile Environments," Autom. Softw. Eng, vol. 9, 2002, pp. 167–186.

[12] D. Tennenhouse, "Proactive Computing," Communications of the ACM, vol. 43, no. 5, 2000, pp. 43–50.

[13] "Silence App," 2015, URL: https://play.google.com/store/apps/details?id=net.epsilonlabs.silence.ads [accessed: 2015-05-13].

[14] "Go Silent App," 2015, URL: https://play.google.com/store/apps/details?id=com.eventscheduler [accessed: 2015-05-13].

[15] "Advanced Silent Mode," 2015, URL: https://play.google.com/store/apps/details?id=com.joe.advancedsilentmode [accessed: 2015-05-13].

[16] "Silent Time," 2015, URL: https://play.google.com/store/apps/details?id=com.QuiteHypnotic.SilentTime&hl=en [accessed: 2015-05-13].

[17] "Auto Silent," 2015, URL: https://itunes.apple.com/us/app/autosilent/id474777148?mt=8 [accessed: 2015-05-13].

[18] Alastair Plumb. Slovakian Violist Lukas Kmit Interrupted By Nokia Ringtone, Incorporates It Into Recital. Huffington Post. [Online]. Available: http://www.huffingtonpost.co.uk/2012/01/23/slovakian-violinist-lukas-kmit-nokia-ringtone_n_1223086.html [retrieved: May, 2015]

[19] D. Zampunieris, "Implementation of a proactive learning management system," in Proceedings of" E-Learn-World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education", 2006, pp. 3145–3151.

 ISBN: 978-1-61208-436-7