A Context-Aware Collaborative Mobile Application for Silencing the Smartphone during Meetings or Important Events

Remus A. Dobrican, Gilles I. F. Neyens and Denis Zampunieris

University of Luxembourg
Luxembourg, Grand-Duchy of Luxembourg
Email: remus.dobrican@uni.lu, gilles.neyens@uni.lu, denis.zampunieris@uni.lu

Abstract—**This study describes a mobile application, i.e., SilentMeet, that uses group-driven collaboration and location-based collaboration for automatically switching smartphones into silent mode during meetings or important events. More precisely, for the first step of the collaboration, a partial agreement algorithm will be used for establishing if a meeting is confirmed by its participants and, for the second round, confirming if the meeting will take place, based on the location of the participants. The application tries to avoid those cases when a meeting is accepted but the participants are not coming to the meeting or when participants do not reply to the meeting invitations but they are still attending the meeting. SilentMeet uses a new technique for exchanging information, for coordinating and for taking distributed decisions, called Global Proactive Scenarios (GPaSs). For executing GPaSs, a rule-based middleware architecture for mobile devices is utilised. GPaSs and the middleware architecture allow developers of collaborative applications to define the actions of their applications in a structured way without having to take care of the communication and coordination of the mobile devices. Also, there is no need for developing a server-side application; all the logic is integrated into GPaSs. Apart the main goal of the application, which is to silence mobile phones during meetings, there are three secondary objectives: a) to provide an collaborative application capable of acquiring contextual information from various devices, b) to check if it is possible to achieve collective reasoning using a rule-based middleware architecture for mobile devices, and c) to validate GPaSs in a real-case example.**

Keywords–Mobile application; Context-Awareness; Location-based collaboration; Collective Reasoning; Proactive Computing; Middleware architecture.

I. INTRODUCTION

This work extends the previous application [1], which was checking if a meeting was potentially validated between the invited users of that meeting. The new application includes an extra checking step, based on location sharing between the mobile devices of the participants, involving multiple rounds of collaboration. Moreover, the application is taking into account contextual information such as the time and the location of the users involved in the collaboration.

Latest studies show a significant increase of mobile devices all over the world [2]. This offers great advantages for developing collaborative mobile applications. However, this brings new challenges like how to handle the high complexity of efficient collaborative mechanisms, how to detect various contexts of users that are continuously on the move or how to automate 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.

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 any more. The demand for services and applications that support communication and collaboration of mobile devices has raised significantly in the past years [3]. 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 [4]. 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 [5], 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 automate 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 [6]. Recently, the possibility of implementing a Proactive Engine for mobile devices was investigated [7]. 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 that has to be executed. More about Proactive Rules and examples with the rules used for this study will be shown in Section V-C.

In order to have a rule-based system capable of executing Proactive Rules on mobile devices, a middleware model was

created for Android-based mobile devices [8]. This represents an important achievement as until now only lightweight basic rule-based engines like [9] and [10] 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) [11]. This method was implemented to give the possibility to the applications to perform collaborative tasks.

In this study, we investigate how a context-aware mobile application, i.e., SilentMeet, which uses a proactive rule-based middleware system, is automatically turning the devices into silent mode if a meeting is detected and confirmed between a predefined group of users and if the location of the users is the same as the location of the meeting, on the same date, at approximatively the same hour.

The rest of the paper is structured as follows. Section II discusses related work relevant to this study. Section III introduces the problem statement and a situation 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 IV contains explanations about SilentMeet's architecture, design and about its way of reaching a global decision based on multiple rounds of collaboration. The Proactive Scenarios that were used for this application and the Proactive Rules that compose them are explained in Section V. Tests on real devices are discussed in Section VI and their results in Section VI-B. And finally, Section VII contains the main conclusions and future work.

II. Related Work

Related work was divided into several categories considered relevant for this study. The first one examines context-aware mobile collaborative systems, where the focus is on the context of groups of users, the second one discusses relevant collaborative middleware architectures, the third one has examples of collaborative mobile applications developed for other fields than the ones that turn the mobile phone into silent mode and the last one contains several examples of mobile applications developed for silencing smartphones in various situations.

A. Context-aware mobile collaborative systems

A key characteristic of mobile collaborative systems, where groups of users perform common activities and have the same interests, is the ability to acquire different contextual information from multiple sources, not only from local, individual sources. The idea is that multiple devices can observe and reason about the same event from different angles. Multiple frameworks were developed to ease the creation of context-aware mobile applications [12] [13] [14], but the aspect of reasoning about the shared contextual information, coming from multiple applications, was not explored. Wang et.al [15] propose a context-aware strategy for collaborative mobile applications based on location. However, the collaboration process is limited as the context information depends only on the near proximity of the participants. Despite a collaborative strategy for sharing context between devices, the authors only provide in [16] a simple integration of the context, which is just added to the knowledge base.

B. Collaborative Middleware Architectures

Numerous studies [5][17][18] 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 [19], 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. MobiSoC [20], a middleware enabling mobile social applications, showed on initial tests indicated that this framework provided good response times for 1000 users for location-based matching and place-based matching.

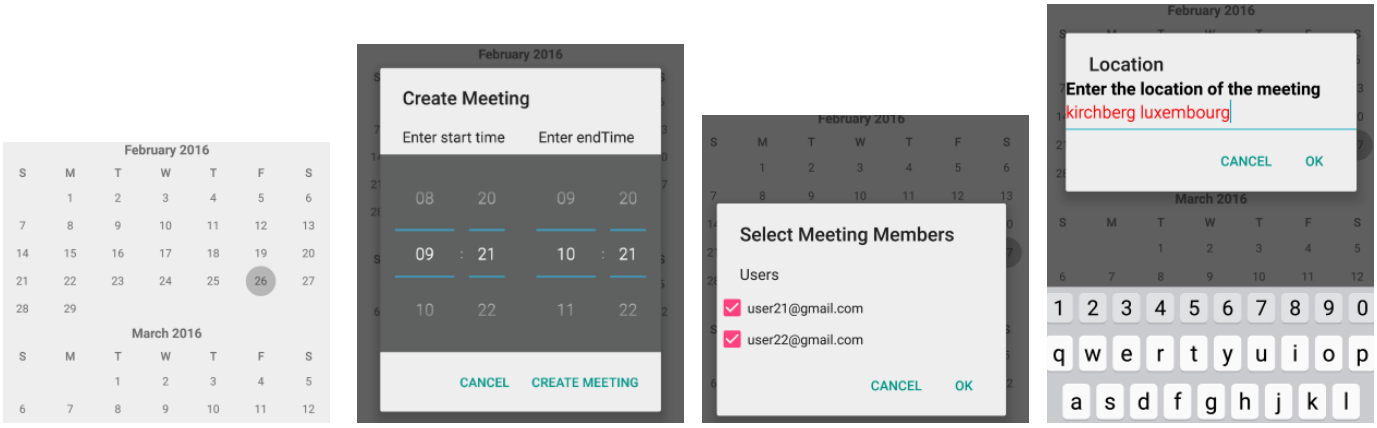
C. Collaborative Mobile Applications

Using the WMP (WatchMyPhone) toolkit, a shared text editing collaborative application was developed in [21] with the help of Mobilis Framework [22]. The proposed toolkit is compressed into a library and can be used by other collaborative applications by including this library into their project. Another domain where collaboration is crucial is represented by mobile-based games. In [23], the authors created a mobile game based on collaborative game play. The game was developed on top of a middleware architecture. However, the whole framework consisted not only from a client side middleware but also from a server side middleware.

D. Applications for silencing the Smartphone

Many mobile applications exist on the market, like Silence[24], Go Silent [25] or Advanced Silent Mode [26], which automatically switch off the sounds of mobile devices based on the user's preferences. These simple applications are focused on one user and perform only local tasks like checking the user's predefined preferences or detecting calendar events. They do not use any kind of collaboration with other devices to make the application smarter.

For example, SilentTime [27] 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 user's input, i.e., a calendar event or 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 [28], which is slightly different from SilentTime because it adds an extra step of verification before muting the user's phone, i.e., it will verify if the user's location corresponds with the event's 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.



(a) The main screen of the SilentMeet (b) Selecting the start and end time of a meeting (c) Selecting users from the same group for sending meeting invitations (d) Entering the location of the Meeting

Figure 1. Creating a meeting

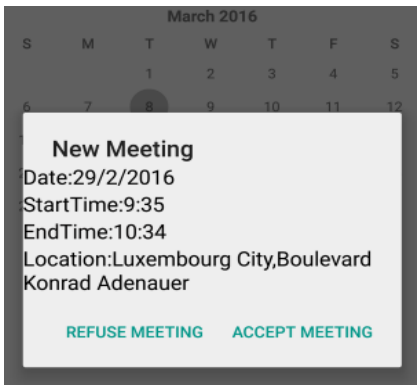


Figure 2. Receiving a meeting invitation

III. Problem Statement

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 [29]. The musician is not only interrupted but he/she could also loose focus and find it difficult to continue. There 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 us imagine the following real-world situation: 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 embarrassing situations.

IV. A rule-based solution - SilentMeet

SilentMeet is a mobile collaborative application that 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, in order to have an efficient collaboration algorithm, part of the user's actions are automated by using Proactive Computing. The main difference between SilentMeet and other applications is that SilentMeet does additional checks, based on collaboration with the other mobile devices, to establish if a meeting is taking place or not. More precisely, it checks, among the possible participants of the meeting, if there are at least 2 users that have accepted to attend the meeting and have the meeting in their calendar, and, finally, on the date of the meeting, it will check the location of the users 15 minutes before the start of the meeting. The additional checks are necessary for trying to avoid cases such as silencing a smartphone even if the meeting is not taking place.

A. The graphical user interface (GUI)

SilentMeet is an application developed for the Android Operating System and consists of a main Activity with a calendar view working as a date picker, laid out in Figure 1a. The basic idea is to provide the user with a simple interface for creating the meeting, i.e., selecting the participants for this meeting, the place where the meeting will take place and the starting and ending hour of the meeting. In Figure 1, an example of creating a meeting using SilentMeet is provided. At the start, the user selects a date from the calendar when the meeting should take place. The date of the meeting should be higher or equal to the current date otherwise the meeting will not be created. Afterwards, a new dialog opens asking the user for the start time and end time of the meeting, as shown in Figure 1b. Again, the start time has to be bigger than the current time if the meeting is on the same day or, if the meeting is on a further day, the start time has to be smaller than the end time.

Then, the user has to select the participants for the meeting. He/she will have to choose from a list of predefined users, as

presented in Figure 1c, i.e., the users that have agreed to be part of the same group for creating future meetings. Additional information about how these groups are created are given in Section IV-B. And finally, before sending the invitation to the other members, the user has to input the location of the meeting, depicted in Figure 1d. The location is given by the user as text, which is then converted into GPS coordinates. These coordinates are stored locally on the phone, and, just before the meeting takes place, they are compared with the current GPS coordinates.

The members selected for the meeting receive an invitation with the date, location, start time and end time of the meeting, depicted in Figure 2. Then, a user can accept, reject or not respond to the invitation by selecting another area on the application's screen. If the invitation is accepted, the response is sent back to the initiator of the meeting. For a meeting to be confirmed there needs to be at least 2 participants that accepted to participate. The initiator of the meeting can cancel anytime the meeting if he/she decides that the meeting should take place only if all the invited members accept the invitation or for other reasons.

B. Grouping the participants for a meeting

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 mobile 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 V-I. 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. In Figure 1c, a user is about to start a new meeting and decides to invite both members of his/her group, i.e., *user21@gmail.com* and *user22@gmail.com*, to this meeting. In this example, these members are part of the same group as the user that creates the new meeting, i.e., *user20@gmail.com*. More about how the users are recognized by the application and how their IDs are handled and the possibility of having multiple groups is explained in Section VI.

C. Middleware model - Proactive Engine for Mobile Devices

The Proactive Engine (PE) for mobile devices is a framework created to support the development of collaborative applications. It contains a middleware architecture capable of executing tasks in the background, of automatically exchanging information with other PEs and of performing actions specific for each application in a structured way. From a technical point of view, the only thing the framework needs from the application developers is a set of Proactive Rules, which is then analyzed, processed and executed. The Proactive Rules represent the structured method of an application of passing instructions to the PE.

1) The Rules Engine: The Rules Engine is the core of the Proactive Engine and is used to process rules provided by different applications [8]. It it composed of two Queues, i.e., two FIFO(first in first out) lists, which contain rules to be executed at each iteration. It is continuously checking for rules to be executed each n seconds, where n is a parameter for establishing the frequency of the checking.

2) Communication between PEs: PEs 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 local server and to Google's Cloud Messaging(GCM) server on the cloud. The GCM server is 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 temporarily available on the network. Message forwarding is done either via WiFi or via 3G/4G, if available. The users of SilentMeet have to have unique identifiers, e.g., in this case unique email addresses, because the PE needs to know where to forward the message or the request for additional information.

V. Proactive Scenarios

A Proactive Scenario is the high-level representation of a set of Proactive Rules that is meant to be executed on the Proactive Engine. It describes a situation and a set of actions to be taken in case some conditions are met. For example, creating a meeting with SilentMeet is achieved with the help of a Proactive Scenario. The set of actions includes defining the date, time, location and the members of the meeting, asking for the members' confirmation, altering the GUI of the application, etc. For each of these actions a Proactive Rule is defined. The Proactive Scenario also consists of defining the order of how the Proactive Rules should be generated and executed by the Proactive Engine. More precisely, a Proactive Rule that asks members for confirming the meeting will only be triggered after a meeting is defined, by another Proactive Rule, on the smartphone of the person that initiated the meeting. Proactive Scenarios are divided into two main categories: Local Proactive Scenarios and Global Proactive Scenarios. An application can have a combination of both types of scenarios depending on its goals.

A. Local Proactive Scenarios (LPaSs)

This type of scenarios is used when defining a situation where only local actions are performed and no collaboration with other devices is needed. They range from simple scenarios that perform simple actions like creating other scenarios when some conditions are fulfilled to complex scenarios, e.g., when the system needs to acquire relevant context information for changing different parameters in order to increase the performance of a PE. Previous examples of LPaSs include supporting students in their learning process through the creation of coaching messages inside their Learning Management System

(LMS) [30] and the creation, maintenance and termination of social groups inside a LMS [31].

B. Global Proactive Scenarios (GPaSs)

On the other hand, a GPaS is a data exchange mechanism, which involves the collaboration of one or more devices. It is based on the data acquisition from multiple sources and it works between all mobile devices with an integrated Proactive Engine. The new generation of interactive applications need collaborative methods that will allow them to find more advanced solutions for addressing existing challenges.

The idea of SilentMeet is that the devices participating in a collaboration process can take decisions based on global information, coming from other PEs, which enhances the local information. Each device is able to make use of the global knowledge that is created by all the devices involved in the collaboration. 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 relevant devices to obtain more precise information about that event by using a particular GPaS.

SilentMeet uses two GPaSs: one for creating and establishing if a meeting will take place and the other one to check, just before the meeting, the location of the users and to decide if they are close to the meeting's location in order to put the mobile phones into silent mode. A meeting is confirmed in two steps: the first step checks if the participants of the meeting have accepted the invitation to the meeting and have that particular meeting in their calendars, and the second step checks the location of the participants to see if it corresponds with the meeting's location, on the exact date, 15 minutes before the meeting is about to start. This algorithm with all the extra checking steps is useful because we want to avoid false positives, i.e., those cases where the meeting is not taking place but the phones are still put into silent mode.

1) Global Proactive Scenario 1.: The purpose of the first GPaS is to create a meeting and establish if a meeting is confirmed by checking with the mobile devices of the other participants. This is only the first step of verifying if the meeting is going to take place. It is necessary for starting the second verification step, i.e., the second 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, e.g., in this case, it will activate the second GPaS. 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. This GPaS allows a decision to be taken without the confirmation of the meeting coming 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.

2) Global Proactive Scenario 2.: The second GPaS is in charge of the second verification step by exchanging the location of participants, if they are close to the meeting's location. So, it is not enough for accepting the invitation when the meeting is created by a user, for example, 1 week before the actual meeting takes place, but there are 2 extra steps to be completed. The first one is that the users that accepted the invitation have to be near the location of the meeting 15 minutes before the meeting will begin and the second one is that they have to exchange their location with at least one other participant that is also near the meeting's location. Only when these steps are fulfilled, the silent mode will be activated. These extra steps of verification are useful for cases when even if persons confirm their attendance at a meeting, they are stuck in traffic, or they had an emergency and cannot attend the meeting, and so, activating the silent mode on their smartphones is not necessary.

```
public abstract class AbstractRule implements
    Serializable{

 @DatabaseField(generatedId = true)
 private long id;

 private boolean activated;
 protected QueueManager engine;

 public AbstractRule(){} // default constructor

 // methods to be implemented
 protected abstract void dataAcquisition();
 protected abstract boolean activationGuards();
 protected abstract boolean conditions();
 protected abstract boolean actions();
 protected abstract boolean rulesGeneration();

 @Override
 public abstract String toString();

 // method used for creating other Rules
 // or for cloning the same rule
 public final void createRule(final
     AbstractRule rule){...}

 // the order of the execution of the methods
 public final boolean execute(){
   dataAcquisition();
   if(activationGuards()){
     this.activated=true;
     if(conditions()){
       actions();
     }
   }
   boolean ret=rulesGeneration();
   return ret;
 }

 // setters and getters
 ...
}
```

Figure 3. The code of the AbstractRule in Java

C. Proactive Rules

GPaSs are composed of sets of Proactive Rules, which are written by the developer and which, among others, contain

a series of instructions. These rules are to be executed by the Proactive Engine when their activation conditions are met, such as, when different events are detected or when they are missing. The initial structure of a Proactive Rule [32] 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*, as depicted in Figure 3, where the code of the **AbstractRule** is provided in the Java programming language used for the Android Operating System. All the other Proactive Rules extend the **AbstractRule**, meaning they have to implement its methods. These methods 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 that starts the negotiation process of SilentMeet.

The SilentMeet application is composed of 2 GPaSs, each being implemented through a small set of proactive rules. These rules are installed together with SilentMeet's user interfaces on each mobile device equipped with a Proactive Engine. Initially, only the Proactive Rules that will continue to clone themselves and be in the Queue at each iteration will be executed by the Proactive Engine. Then, all the rules can be activated, if their execution conditions are met. One of the rules that is executed at the beginning by the PE is called **RegisterToServerRule** that registers the user on the GCM server, if not already registered. This will give the user a unique ID, which is then used in the communication with the other PEs.

D. Proactive Rules that compose the first GPaS

GPaS1 is composed of 4 Proactive Rules, i.e., **R011**, **R021**, **R012** and **R022**. **R011** is one of the rules that is running from the beginning, when the application is installed, and is checking for new meetings in the local database. The code of rule **R011** is shown in Figure 4. The PE executes an Iteration each 5 seconds, so, **R011** will be checking each 5 seconds for a new meeting. When creating a meeting, as seen in Figure 1, SilentMeet registers the meeting's location, date, start time, end time and the persons invited to that meeting. The status of the meeting, after it was created locally, is *pending* and *unsent*. **R011** checks for all the pending unsent meetings, this step being part of the **data acquisition** method of this rule, and, only if such meetings are detected, the **actions** method will be activated. Inside this method, an invitation will be sent to the users selected to attend the meeting. The invitation will contain all the meeting's details like its location, start time, end time, date and members. The name of rule to be activated on the receiving PEs is included among the parameters when the message is sent to the receiving PEs. And so, rule **R021** will be activated on the devices of the receivers. For example, if user1 decides to create a meeting and invite user2 and user3 to that meeting, on the devices of user2 and user3 the PE will activate rule **R021**.

When rule **R021** gets activated, it means that the receiver of the message is invited to a new meeting. In its **data acquisition** phase it looks in its own calendar if there is no other meeting

```
@DatabaseTable(tableName="R011")
public class R011 extends AbstractRule{

   // local parameters
   private long startTime, date;
   private boolean newMeeting = false;
   private List<Meeting> meetings;

   @Override
   protected void dataAcquisition() {
      // if a new meeting is detected
      newMeeting = engine.getLpeDBWrapper().
          isNewMeeting();
   }
   ...
   @Override
   protected boolean conditions() {
      return newMeeting;
   }

   @Override
   protected boolean actions() {
      meetings = engine.getLpeDBWrapper().
          getListOfPendingMeetings();
      ArrayList<Object> p; // paramsToSent

      for(Meeting m : meetings) {
         p = new ArrayList<Object>();
         p.add(m.getMembers());
         p.add(m.getDay());
         ...
         p.add(m.getLocationLongitude());
         p.add(engine.getContext().
             getResources().
             getString(R.string.mail));

         ArrayList<String> deviceIDs = new
             ArrayList<String>
             (Arrays.asList(m.getMembers()));
         try {
            if(!engine.getLpeDBWrapper().
                meetingRequestWasSent
                (m.getId())){
               engine.sendMessage("R021", p,
                   deviceIDs, 100);
               SentMeeting sentMeeting = new
                   SentMeeting(m.getId());
               engine.getLpeDBWrapper().
                   save(sentMeeting);
            }
         } catch (Exception e){
            Log.e("R011", "message was not
                sent");
         }
      }
      return true;
   }

   @Override
   protected boolean rulesGeneration() {
      createRule(this);
      return true;
   }
   ...
}
```

Figure 4. Proactive Rule R011 in Java

```
public class R024 extends AbstractRule{
    ...
    @Override
 protected boolean actions() {
      AudioManager am = (AudioManager)
          engine.
          getContext().getSystemService
          (Context.AUDIO_SERVICE);
        //For Silent mode
     am.setRingerMode
         (AudioManager.RINGER_MODE_SILENT);
      return true;
    }
    ...
}
```

Figure 5. Devices used for testing SilentMeet, in the process of receiving an invitation for a meeting

on that specific date and time and if this invitation has not already been accepted. If these conditions are satisfied, then this rule will trigger a pop-up dialogue on the mobile phone of this user to ask him/her if he/she accepts to attend this meeting, as seen in Figure 2. The operations are part of the **actions** phase of the rule. This rule does not generate other rules and does not clone itself. When receiving the invitation inside the pop-up, the user has 3 options: accepts the invitation, rejects the invitation or does not respond to the invitation by changing the application or by clicking on another part of the screen. In case he/she accepts the meeting, rule **R012** gets activated.

Even though **R012** is one of the rules which is executed by the PE at each iteration, it only gets activated when the conditions are true, i.e., the user accepts or rejects a meeting. The immediate effect, if the conditions are true, is to send the response to all the users invited to attend the meeting. If the answer of the user is positive and he/she accepts to join the meeting, then, at this particular moment, there are at least 2 persons that accepted to attend the meeting. In case the answer is negative, the device of the same user will register the meeting as refused and even if the meeting will still take place with the other participants, the devices of this user will not be switched into silent mode.

The receiving devices activate rule **R022** that gets as parameters the answer of a user with regard to a specific meeting invitation. If the answer is positive and the meeting is confirmed by at least 2 persons, the second GPaS will be activated. If the answer is negative, the device of the initiator of the meeting still waits until all the answers from the invited members will be received. Until then, the meeting will be in *pending* mode. If all the answers are negative or part negative and part unanswered before the meeting starts, the meeting will be considered as *canceled*.

E. Proactive Rules that compose the second GPaS

GPaS2 is composed of 3 Proactive Rules, i.e., **R013**, **R023** and **R024**. **R013** is only activated when a meeting has been created and accepted by at least 2 participants. It will check the current time on the device, and, if it is equal or less but not more than the meeting's start time minus 15 minutes, it will

```
{
  "PARAMETER_TYPES":[
    "String[]",
    "Integer",
    "Integer",
    "Integer",
    "Integer",
    "Integer",
    "Integer",
    "Integer",
    "Double",
    "Double",
    "String"
  ],
  "PARAMETER_VALUES":[
    [
      "user20@gmail.com",
      "user22@gmail.com"
    ],
    29,
    3,
    2016,
    19,
    6,
    20,
    6,
    49.6278694,
    6.153422,
    "user21@gmail.com"
  ]
}
```

Figure 6. An example of a JSON message that is passed between R011 and R021, when a meeting is created

start to check for the location of the device. If the location also corresponds to the location of the meeting, then the condition for executing the rule's **actions** are met. These actions include sending a message to the other participants to confirm the device's presence at the meeting's location. After sending the message, the device of this user that activated **R013** waits for receiving at least one message from another PE of a participant in order to activate the last rule, i.e., **R024**, which turns the smartphone into silent mode. Checking for the location of the user every 5 seconds consumes a lot of battery, so, this action is performed only when the current timestamp approximatively corresponds to the meeting's timestamp.

Upon receiving the message from one user that is close to the meeting's location, the PE of the other participants activate **R023**. This rule tells the local PE that there is at least 1 person attending the meeting and so, has permission to switch the smartphone into silent mode if the local PE is close to the meeting's location. If this last condition is carried out then the last rule is activated, i.e., rule **R024**.

The last rule of GPaS2 is in charge of finally silencing the mobile phone during that meeting. The only way this rule is executed by the PE is to get through all the previous collaboration steps of both GPaSs and to fulfill all the necessary conditions of each rule. The command for silencing the device is given in the **actions** phase of the rules, as seen in Figure 5. So, a user that did not reply with yes or no for attending

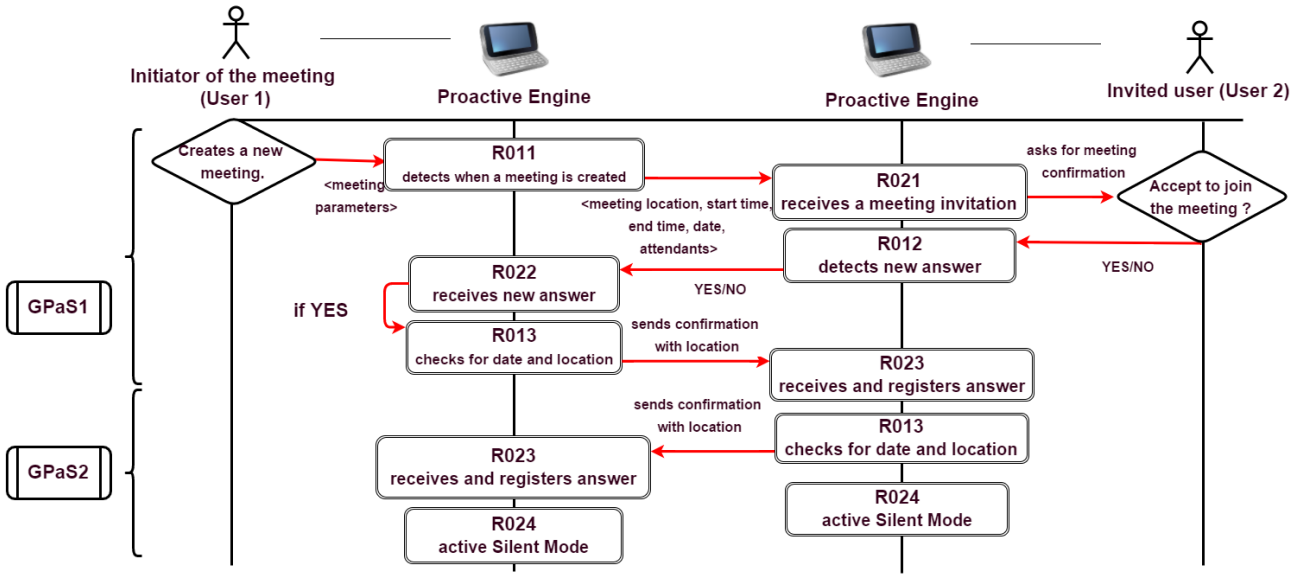


Figure 7. Sequence diagram with the collaboration steps of SilentMeet

the meeting, can have his/her mobile phone switch into silent mode if his/her device are close to the meeting's location, on the same date and same hour as the meeting. SilentMeet considers that by fulfilling these conditions the device is very likely to participate at that meeting, even though it did not provide a precise answer. This case includes a hybrid algorithm for establishing if a meeting will take place or not. The existing algorithms either check for an entry in the calendar or, more advanced applications, just check for the location of the current user but not the other users' location.

F. Cyclic Proactive Rules

A Proactive Rule can have the property of being *cyclic* if it continues to clone itself and gets executed by the PE at each iteration. For example, rules **R011**, **R012** and **R013** are *cyclic* because they need to continuously check for new meetings, for new answers from the users or for matching dates and locations of meetings. Cyclic rules can be generated by other rules and do not have to run from the beginning, when the PE starts.

G. Non-Cyclic Proactive Rules

Rules such as **R021**, **R022**, **R023** and **R024** are non-cyclic because they contain specific actions that need to be performed only once. They are usually triggered by other rules and are part of a chain of rules. Multiple examples of chains of rules are given in Figure 7. For instance, one chain starts with rule **R011** and ends with **R021**, which receives an invitation to attend a new meeting.

H. Message Exchange between Proactive Rules

Proactive Engines exchange information between each other with the help of JSON messages. The messages can contain commands to activate certain Proactive Scenarios or they can contain just simple context information. Figure 6 shows the content of a JSON message that is exchanged between Proactive Engines. More exactly, when a meeting is created on the device of the user with the email address *user21@gmail.com*, rule **R011** gets activated and sends a message to *user20@gmail.com* and to *user22@gmail.com*. The devices that get the invitation to the meeting activate rule **R021**, which receives precise data about the date, start time, end time, location, sender and list of invited members of the meeting.

I. Collaboration 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 collaboration. These rounds of collaborations are depicted in Figure 7 with the help of a sequence diagram. Moreover, it is shown how rules are activated by other rules. This example shows what will happen on the PE from the beginning of GPaS1, when a meeting is created on one smartphone, until the end of GPaS2, when the meeting is confirmed and the devices are silenced. The first GPaS can be activated, for example, 1 week before the meeting actually takes places but the second GPaS needs to wait until the same date and approximatively the same hour of the meeting to get activated. A user can receive multiple invitations in the same time and does not have to worry about how they will be handled. This is done automatically by the Proactive Engine. The collaboration process depends on the communication of PEs, which depends as well on the connectivity setting of each mobile device. The PE performs also error checking and handling in case a message is lost somewhere in the network and no answer is received from other PEs.

VI. Tests

Tests were conducted locally at our university on 3 different devices: a Samsung Galaxy Note 3 and two Samsung Galaxy S6, as shown in Figure 8. All 3 devices use an Android

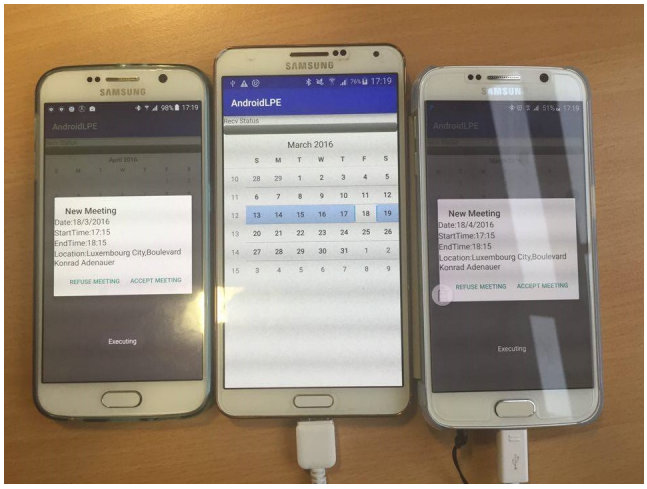


Figure 8. Devices used for testing SilentMeet, in the process of receiving an invitation for a meeting

operating system and have SilentMeet on top of the Proactive Engine middleware installed in order to be able to execute rules and collaborate with each other. The devices were part of a predefined group of 3 participants with the following email addresses used as unique identifiers: *user20@gmail.com*, *user21@gmail.com* and *user22@gmail.com*. During the tests, all 3 devices were connected via WiFi to the same network. Initially, all the devices had their sound turned on.

In the first series of tests, the user with the email address *user20@gmail.com* and using the Samsung Galaxy Note 3 was the initiator of a meeting and created it on SilentMeet's local calendar. The meeting was set to happen after 10 minutes of its creation time, on the same date, in the same location as all the devices, i.e., the campus at our university. The invitation was displayed on the screen of the 2 other mobile phones and, after the meeting was accepted by both guests, it was marked in the calendar of SilentMeet. The devices started immediately to check their locations, compared it to the meeting's location and shared it with the other guests, as the start time of the meeting was very close to the current time, i.e., less then 15 minutes difference. All 3 devices were silenced when the meeting started.

The second series of tests happened in the same conditions as the first tests except with one minor detail: one of the invited users did not accept or reject the meeting proposal. However, the minimum of 2 persons that accepted the invitation was reached and so, all the devices had activated GPaS2, which checked their locations before the start of the meeting. Because all the other conditions were accomplished, the 3 devices were silenced again when the meeting started.

For the third series of tests, the 3 users that were part of the meeting proposal accepted the invitation but only one user was at the same location as the location of the meeting, i.e., the user with the email *user20@gmail.com*. The device of this user did not get a location confirmation from the other 2 users so it did not switch into silent mode. Neither the 2 devices of the 2 other users.

A. Measurements

The main goal was to check if the application behaved as expected in the most common cases, e.g., when all the users confirmed their presence at a meeting and their location is the same as the meeting's location when it started, as well as the unusual situations. These unusual situations include not providing an answer of participating to a meeting but still attending that meeting, accepting the meeting invitation but not coming to the meeting or being the only one present at a confirmed meeting where nobody else is present.

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.

VII. CONCLUSION AND FUTURE WORK

In this paper, we demonstrate that it is possible to easily design and implement a context-aware collaborative application on top of a rule-based middleware engine and with the help of Proactive Computing, more precisely, by using Global Proactive Scenarios. SilentMeet is able to detect and acquire relevant context-information about calendar events, to use a collective reasoning algorithm to establish if a meeting will take place or not and to take decisions of silencing the smartphone, based on the shared locations of the users. Furthermore, the location sharing process is handled very efficient in order to reduce the unnecessary battery consumption.

At the same time, several parts of the collaboration process were automated and the user's involvement reduced only to the most important operations. SilentMeet reduces the possibility of having meetings in the calendar that do not take place any more or which are canceled by the other participants. The smartphone turns into silent mode only when multiple conditions are met, reducing thus the risk of having the smartphone on mute when not attending any event. With only two GPaSs, composed of seven Proactive Rules, it is enough to achieve SilentMeet's goals.

Short term future work includes extending the application for checking for meetings not only in the calendar provided by the application but on the local calendar of the smartphone together with the calendar of other applications that are intensively used, such as Google's Calendar or the Outlook Calendar. Another point would be to enhance SilentMeet to allow the users to create, with the help of an additional GPaS, their own groups of people for meetings that happen more regularly. Long term developing more complex collaborative applications and other Global Proactive Scenarios on top of the Proactive Engine. These applications could have different application fields such as tele-medicine, transportation or e-Learning, where collaboration is a key aspect.

REFERENCES

[1] R.-A. Dobrican, G. Neyens, and D. Zampunieris, "Silentmeet-a prototype mobile application for real-time automated group-based collaboration," in Proceedings of the 5th International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2015). IARIA, 2015, pp. 52–56.

[2] Kate Dreyer. Mobile Internet Usage Skyrockets in Past 4 Years to Overtake Desktop as Most Used Digital Platform. comScore. [Online]. Available: https://www.comscore.com/Insights/Blog/Mobile-Internet-Usage-Skyrockets-in-Past-4-Years-to-Overtake-Desktop-as-Most-Used-Digital-Platform (2015)

[3] 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]

[4] 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]

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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

[10] 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.

[11] 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.

[12] E. Benítez-Guerrero, C. Mezura-Godoy, and L. G. Montané-Jiménez, "Context-aware mobile collaborative systems: Conceptual modeling and case study," Sensors, vol. 12, no. 10, 2012, pp. 13 491–13 507.

[13] E. Williams and J. Gray, "Contextion: A framework for developing context-aware mobile applications," in Proceedings of the 2nd International Workshop on Mobile Development Lifecycle. ACM, 2014, pp. 27–31.

[14] S. Elmalaki, L. Wanner, and M. Srivastava, "Caredroid: Adaptation framework for android context-aware applications," in Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. ACM, 2015, pp. 386–399.

[15] W. Wang, J. Gu, J. Yang, and P. Chen, "A group based context-aware strategy for mobile collaborative applications," in Advanced Technology in Teaching. Springer, 2012, pp. 541–549.

[16] L. Zavala, R. Dharurkar, P. Jagtap, T. Finin, and A. Joshi, "Mobile, collaborative, context-aware systems," in Proc. AAAI Workshop on Activity Context Representation: Techniques and Languages, AAAI. AAAI Press, 2011.

[17] 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.

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

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

[20] A. Gupta, A. Kalra, D. Boston, and C. Borcea, "Mobisoc: a middleware for mobile social computing applications," Mobile Networks and Applications, vol. 14, no. 1, 2009, pp. 35–52.

[21] S. Bendel and D. Schuster, "Watchmyphone - providing developer support for shared user interface objects in collaborative mobile applications," in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on, March 2012, pp. 166–171.

[22] R. Lübke, D. Schuster, and A. Schill, "A framework for the development of mobile social software on android," in Mobile Computing, Applications, and Services. Springer, 2011, pp. 207–225.

[23] F. Klompmaker and C. Reimann, "A service based framework for developing mobile, collaborative games," in Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology, ser. ACE '08. ACM, 2008, pp. 42–45.

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

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

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

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

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

[29] 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]

[30] R. Dobrican and D. Zampunieris, "Supporting collaborative learning inside communities of practice through proactive computing," in Proceedings of the 5th annual International Conference on Education and New Learning Technologies. IATED, 2013, pp. 5824–5833.

[31] D. Shirnin, S. Reis, and D. Zampunieris, "Design of proactive scenarios and rules for enhanced e-learning," in Proceedings of the 4th International Conference on Computer Supported Education, Porto, Portugal 16-18 April, 2012. SciTePress–Science and Technology Publications, 2012, pp. 253–258.

[32] 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.