Moving Towards Distributed Networks of Proactive, Self-Adaptive and Context-Aware Systems: a New Research Direction?

Remus-Alexandru Dobrican and Denis Zampunieris
Computer Science and Communication Research Unit
University of Luxembourg
Luxembourg, Grand Duchy of Luxembourg
Email: {remus.dobrican, denis.zampunieris}@uni.lu

Abstract—**Instead of being static and waiting passively for instructions, software systems are required to take a more proactive approach in their behaviour in order to anticipate and to adapt to the needs of their users. To design and develop such systems in an affordable, predictable and timely manner is a great software engineering challenge. Even though there have been notable steps for modelling self-adaptive and context-aware systems, there is still a lack of a generic model agreed by the research community for developing smart applications. The goal of this study is to explore the idea of having multiple networks of proactive context-aware adaptive systems working together for achieving common goals. To support our vision, we introduce a context-aware self-adaptive software model for mobile devices capable of learning from the user's behaviour by using Proactive Computing. The novelty comes from the possibility of developing smart applications that would benefit from the proposed properties. Moreover, we discuss a motivating scenario that led to this work and propose a case study where a collaborative e-Learning application is implementing our model.**

Keywords-smart applications, self-adaptive systems, context-aware systems, proactive computing, distributed networks.

I. INTRODUCTION

This paper is an extension of the work-in-progress presented in [1], where we introduced a vision showing the tendency of moving towards various networks of context-aware, proactive and self-adaptive systems. In this extended version, we support our vision by analyzing the latest proposed middleware, architectures and applications that try to incorporate the mentioned properties, by providing a motivating scenario and by proposing a model for software systems capable of integrating all the above properties.

The demand for devices and applications that are able to adapt their behavior at run-time, as a response to the increasing demands of users, has risen considerably in the last couple of years [2]. Giving instructions to complex software systems is becoming quite a difficult task for the users, as it requires their continuous involvement, a set of advanced technical skills and a lot of knowledge about the system. As a consequence, our model is leading the users towards new ways of interacting with smart systems that will be able to perform a variety of automated tasks on users' behalf. Three main properties are to be distinguished when speaking about systems that dynamically adapt themselves according to the context variation or the requirements change: *Self-Adaptation*, *Proactivity* and *Context-Awareness*. Modeling such systems is a difficult task because important aspects like the user's needs, the settings of the environment in which they are deployed and the all the other requirements have to be taken into account. There is no standard model agreed upon by the research community, which could serve as a common basis for developing smart applications.

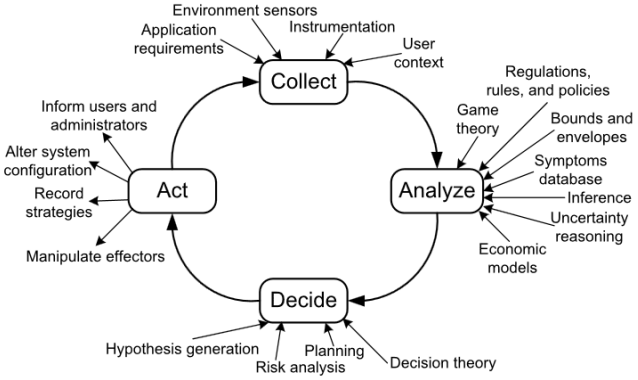

Figure 1. Autonomic control loop [4]

Self-adaptation in software systems comes in many different aspects. Self-adaptive mechanisms provide the necessary means to make changes either at an architectural level or at a behavioral level. Systems that possess this property can be characterized by their operating mode that permits them to fulfill easily their goals in a modified context.

Feedback loops provide an architectural solution for self-adaptation. In [3], the authors indicate that feedback loops usually include four key activities: *collecting*, *analyzing*, *deciding* and *acting*. These activities are essential for achieving self-adaptability, context-awareness and proactivity. In Fig. 1, a generic model of a unidirectional feedback loop is given. It shows the inputs or the outputs of each state but the data flow between the states is omitted. During the first stage of the feedback cycle, the collection of data, relevant context information can be acquired by the system from the environmental sensors and from other internal and external sources. Then, the data is analyzed by the system according to its policies and constraints, while taking in account the user's preferences. The process of transforming raw data into relevant context-information is

quite problematic and complex because of many errors and broken sensors. A solution for this particular problem was proposed in [5]. Once the system reaches the third phase it will reason if future actions are needed or the system can get back in its initial state of collecting information. If future actions are required, then the system enters in the fourth state, were the actual adaptation happens. And here comes one of the biggest challenges: should the system have enough authority to perform adaptation in all cases?

The above feedback loop was inspired from IBM's four-stage cycle for autonomic computing called MAPE-K [6]. One of the main differences is that the MAPE-K model uses a knowledge base shared between all the stages.

Context-aware systems open new dimensions and opportunities for developers to create smart applications, especially for mobile devices. These systems are designed to continuously analyzing contextual information, which is a key feature for determining the occurrence or the lack of events. When a system is aware it means it acquired knowledge about the surrounding environment by its own means. Awareness refers to the state of being aware in which a system can be found. Various sensors capture the data that creates context information. They can be classified in three categories: physical, virtual and logical [7]. Physical sensors are the most frequent used type of sensors and they can provide data about location with the help of GPS, GSM, RFID tags, optical data from the cameras and IR sensors, motion data from accelerometers and gyroscopes, and ambient data collected from thermometers and barometers [8]. Virtual sensors can capture context data from applications and services including operating system events, application data and network events. Logical sensors combine data received from the other two types of sensors and they provide higher-level context information.

Events play a central role in the lifecycle of software systems. They range from simple request for different services to serious incidents that prevent the well functioning of a system. Events can be divided into three main categories: foreseen (*taken care of*), expected (*planned for*) and unexpected (*not planned for*) [9].

Tennenhouse [10] first introduced Proactive Computing as a new mode of operation that was crucial for moving towards human-supervised computing. The essential features of proactive systems, as seen in [11], are taking decision for their users and acting on their own initiative. Proactive Computing is a solution for foreseeable events, while Context-Awareness and Self-Adaptation handle unforeseen events, which are seen as deviations from normal situations.

The contribution of this paper is three-fold. First, it examines the most relevant work in several research fields like Proactive Computing, Context-Awareness and Self-Adaptation, for pointing out current research direction, what is missing and what are the next steps to be taken. Second, it offers an middleware architecture for supporting smart mobile applications, capable of performing automated tasks for the user, of analyzing large quantities of data and of making decision in different contexts. And third, it provides an analysis of a distributed network of mobile proactive systems that are implementing our model.

The rest of the paper is organized as follows: Section II provides an overview of related work relevant to our research, in Section III we describe a motivating scenario that lead to this work, Section IV describes the main components and the characteristics of a Proactive Engine, Section V investigates the possibility of having multiple distributed networks of Proactive Engines, Section VI provides an example of a smart application based on our model, Section VII proposes multiple domains where applications using our model could be developed, Section VIII contains a brief implementation overview and, in Section IX, we conclude and we emphasize the potential of Proactive Engines.

II. RELATED WORK

In this section, we focus mainly on previous work done in the area of *Proactive Systems*, *Context-Aware Systems* and *Self-Adaptive Systems*, which indicates an intensive effort to develop middleware, architectures, frameworks and prototype applications that would serve in dynamic environments. These research initiatives show the growing interest for developing a new kind of intelligent systems that will be able to perform complex tasks. Our contribution in this paper has been to find a platform that unifies multiple research efforts in the above research fields.

A. Proactive Systems

The concept and the structure of the first Proactive Engine (PE) were created in 2006 [12]. It was among the first systems specially conceived to use Proactive Computing for achieving its goals. The PE was designed as a complex mechanism for running Proactive Rules. A Proactive Rule is a structure conceived to perform specific actions in case a special situation was detected or in case of the lack of an event. The detection of students that did not submit their online assignment and the notification of their professor as a consequence, is a concrete example of a rule, which was used in a real-case scenario, when the initial Proactive Engine was deployed aside a Learning Management System (LMS) [13]. Results showed that major limitations of a LMS such as the restricted interaction and limited collaboration between learners and educators inside courses could be overcome with the help of a Proactive Computing [14].

In [15], a mechanism for creating, organizing and developing Communities of Practice (CoPs) inside a LMS using Proactive Computing was proposed. Three types of virtual CoPs were developed: for students coming from the same country, for students living in the same cities and for students enrolled in the same study program. The LMS was capable of automatically enrolling students in these communities, creating tools like forums, chat and folders for stimulating collaboration inside the CoPs and adjusting the size of the CoPs. The Proactive System performed these

operations without any explicit command from the administrator of the LMS.

Previous works [15], [16], focused until now on applying Proactive Computing on a single system, thus exploring only the possibility of having only one centralized Proactive System. But a centralized solution can become quite fast non-scalable in many scenarios where a Proactive Engine handles a big number of devices and applications. Using a single Proactive System presents certain limitations and cannot benefit from the advantages that multiple Proactive Systems, working together and collaborating, would have to offer.

The possibility of having multiple networks of connected Proactive Engines, which are able to communicate, to create a shared knowledge base, to perform joint actions and to learn from each other, was not yet explored.

B. Context-Aware Systems

Even though there is an intensive effort to define a standard context-aware model for developing smart applications [17]-[20] there is a lack of precise guidelines on how to integrate context-aware mechanisms into these applications. A basic approach for modeling the adaptation requirements, SOTA, is presented in [21]. In particular, it was conceived to enable early requirements verification, the identification of knowledge requirements and the most suitable self-adaptive patterns. *Cinemappy*, a location-based application that is able to compute contextual movie recommendations [17], is a simple example how contextual information related to the temporal and spatial position of the user is exploited for obtaining personalized recommendations results for each user. Pushing relevant application to mobile devices, according to the user's location, using a context-aware architecture called *MoBe* [18] is another example for addressing cases where the location of the devices is constantly changing. *MoBe* allowed data exchange and joint actions for reaching a common goal between the cooperating devices. The design process of *OnRoute*, a context-aware mobile travel application for handling navigation in public transportation was studied in [19]. A smart way of using location information for facilitating navigation was incorporated in the application, which was able to learn from the user's behavior and propose other route alternatives. *Museum Guide,* an indoor location-based, context-aware and video on demand application [20], was able to detect if a user was moving, and, based on this contextual information, to play a certain video, related to the nearby exhibit.

Furthermore, the role of context-information for improving collaboration between mobile devices by delivering relevant information to the participants involved in the cooperation process is investigated in [22].

C. Self-Adaptive Systems

In many disciplines, ranging from Artificial Intelligence to biology, self-adaptation of software systems has been studied and only recently the software engineering community has realized the major role this property plays in developing and implementing software systems that are able to comply with important changes in the environment and in the requirements [23].

In the latest research literature many approaches study adaptive systems. However, a great amount of them are concentrated on architectural adaptation [24][25][26], parametric adaptation [27][28] or implementation adaptation [29]. More precisely, they focus on changing the internal structure of a software system, on modifying the necessary parameters of the system and on changing the implementation of different components of a software system without changing the interface. In comparison, we propose a model that allows developers to reason about adaptation at a higher level of abstraction, without having to take care of the low-level implementation details.

Multiple studies investigate how to apply adaptation mechanisms for mobile applications. For example, in [30], a battery-efficient architecture was proposed for building battery-aware applications. More recent work [31] studied a mechanism for an effective adaptive real-time multimedia content delivery for smartphones using a hybrid mobile application development platform. A mobile learning case study was discussed based on the hybrid learning application.

D. Context-Aware Adaptive Systems

Event though many researchers are treating Context-Awareness and Self-Adaptation as two separated research fields there have been numerous studies aimed to provide applications, tools and frameworks that can incorporate and provide both properties.

In [32], the authors acknowledge that embedding mechanism for achieving context-awareness and self-adaptation in pervasive systems is crucial. They propose a model-driven engineering approach able to integrate such mechanisms. A separated context model but related to the adaptive model is proposed in [33] for modeling and realizing context-aware adaptive software systems. Management context is differentiated and separated from operational context in order to obtain run-time adaptation. A tool, CAST, was recommended for modeling the system's implementation and of verifying and validating the system's context-aware adaptive behavior.

In order to address problems like the high complexity of modeling the behavior of context-specific application and of structuring application code for efficient switching between various functions of the system, a mobile middleware was discussed in [34]. This middleware system was created on top of the Android operating system as tool for building adaptive, context-aware ubiquitous mobile applications. A prototype application, i.e., a context-aware Instant Messenger, was set up using the proposed middleware and compared with previous versions of Instant Messenger. A generic adaptation model was analyzed in [35] for helping

designers to develop ambient assisted living applications that enabled elderly people to live longer in their residential homes. Furthermore, a context-aware and adaptive learning scheduling framework for supporting the student's daily routines was introduced in [36].

III. MOTIVATING SCENARIO

The following scenario has motivated the need of having groups of proactive context-aware adaptive systems capable of communicating and sharing information in a transparent way. When scheduling a meeting, the users have to give many complex instructions for arranging the exact time, location and group of participants. They have to be involved in each steep of the selection process, which requires certain technical skills. Also, they have to manually check other applications like the integrated agenda or calendar to be sure there are no overlapping activities. Below, the motivating scenario is presented with the help of several scenes.

Scene 1: A professor, the head of a research department at a university, receives an email from the human resource department about the budget for the next year. As a consequence, he/she would like to arrange a meeting with his colleagues, also professors from the same research department, for discussing and who would the budget be spited to cover the expenses of each professor and his/her team of assistants for research materials, for traveling purposes and for participating at different conferences/workshops.

Scene 2: He/she then uses the smart meeting application on his/her smartphone for scheduling a meeting with the other professors. Inside the application, he/she can select different users or different target groups of users for proposing a meeting. These groups are either arranged manually or automatically, based on previous activities performed together with various users. Then, he/she would select the period when he/she would like for the meeting to take place. For instance, the time interval would be the next 2 days, starting from the moment of the proposal. At this stage, the application would start the communication process with the other smartphones for finding a possible date and time for the meeting.

Scene 3: The application starts to perform internal and external processes for finding relevant information that would allow it to find a possible solution for the meeting. Internally, the application would look for information on other applications like the integrated calendar application or the Google Calendar, in case the users have a Gmail account, or in the Outlook Calendar where the users could have additional information about future meetings. Externally, requests are sent by the application to the mobile phones and tablets of the other participants. These requests start to be processed locally by the other devices that start to look for relevant context information that would allow them to find if their users are available or busy in the interval of time proposed by the initiating device. And so, a negotiation process starts between devices for finding the best solution to arrange the meeting.

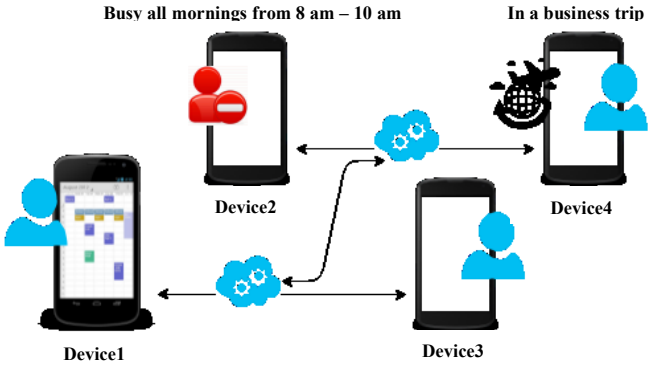

Figure 2. Establishing a meeting between smartphones

Scene 4: Devices start to share information about the potential time slots that are available for the meeting as illustrated in Fig. 2. For instance, the application detects that the user of device 4 is at a remote location for the next 24 hours and cannot participate in any meeting close to the location of the other users. The user of device 2 is busy all the days of the week in the morning because he/she has to give classes from 8 am until 10 am. The two other users do not have anything scheduled in the next 2 days and are available for a meeting. After several rounds of negotiation, they find a free time slot in the next day where everyone can attend the meeting.

Scene 5: At the end of negotiation process, after an optimal solution is found and agreed upon by all the devices, a confirmation request is sent to all the users for making them aware of the meeting and to get their final approval. If there are users that do not agree with the solution then new rounds of negotiation begin until either another solution is found or the user that initiated the meeting stops the whole process.

Scene 6: If an agreement was reached about the day and the hour of the meeting, the application can start to finalize scheduling the meeting by proposing different locations. Locations can be either introduced manually by the users or can be proposed by the application based on multiple factors like previous locations or available conference rooms on the campus, information that can be found on the research department's server.

Possible issues can arise from the fact that some users could not be reached because their devices are either offline or they are turned off. In these cases, where a universal solution involving all the participants could not be found, the application either proposes a partial solution, for a meeting with the users that were found available, or the meeting is postponed until all the users will be available. In both cases, the user that initiated the meeting procedure will be notified about the outcome of the negotiation process and he/she could take additional measures, and the other participants would receive mails and/or SMS to be informed that they were requested to join a very important meeting.

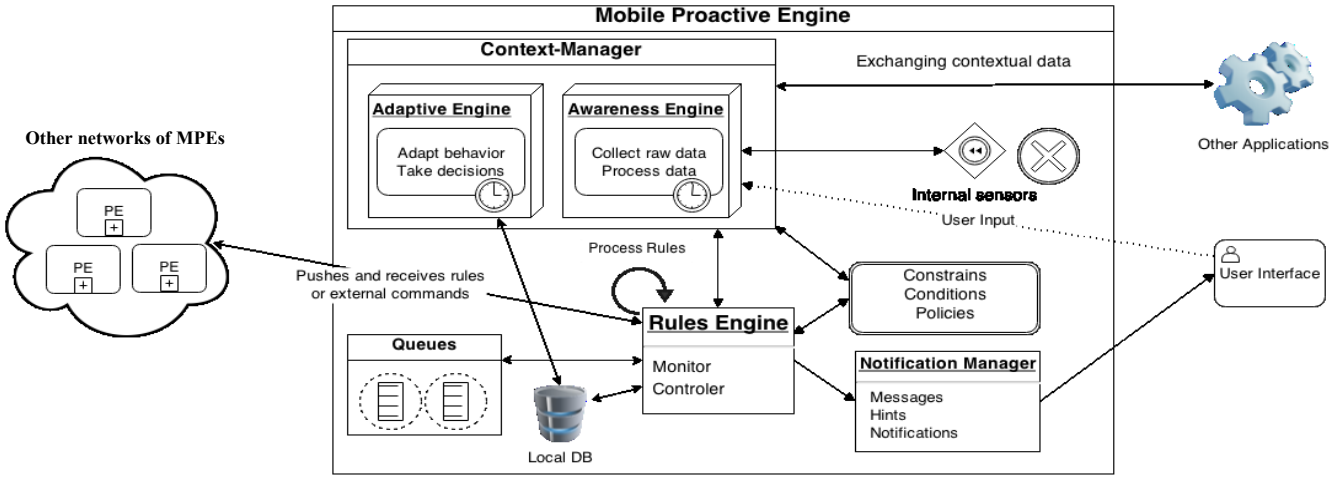

Figure 3. The structure of a Proactive Engine for mobile devices

Our goal, in this scenario, was to present an application capable of performing automated actions on behalf of the user, e.g., automatically search for possible free time slots for organizing a meeting. Therefore, the following model proposed in the next section allows the creation of such an application that implements all the above features.

IV. THE APPROACH

We propose a new version of the Proactive Engine for mobile devices, called Mobile Proactive Engine (MPE), where processes are divided between the sub-parts of the model. Before, Proactive Rules were taking care of data acquisition, activation guards, conditions, actions and rules generation. In the current model, each step is assigned to a specific component. A major benefit of separating these processes is that they are handled by structures that are focusing only on particular tasks.

In order to develop a proactive context-aware adaptive system, an infrastructure that combines and uses all three properties is required. The MPE is an advanced mechanism that could be easily integrated into new software systems because it provides means for gathering data from the internal and external sensors, for detecting context changes, for processing and modeling contextual information, for executing adaptive tasks and for providing an adequate system behavior in any situation. The term "sensor" refers not only to the hardware parts being able to sense but also to the various data sources that may give contextual information.

Thus, the architecture of a MPE is composed of a set of interconnected components, including a Context-Manager, a Rules Engine connected to a set of Queues and a local database, and a Notification Manager, as seen in Fig. 3. These components are able to communicate, sending and receiving messages or specific commands from the other components. For example, a Proactive Rule in the Rules Engine may require some additional information for responding to a situation, and would be able to activate the Awareness Engine. Then, additional information would be acquired from the sensors and, after verifying local constraints and conditions, it would be send back to the Proactive Rule that asked for additional data. The components of a MPE cannot perform structural adaptation processes. They are oriented for helping the MPE to perform a behavioral adaptation, where the initial functionalities of the MPE can be changed.

A. The Context-Manager

This component is very important, as it behaves as a filter for the majority of the data acquired by the MPE. The need of a Proactive Filter is justified by the huge amount of unnecessary data gathered by a MPE. Data coming from different sources can contain many errors or mistakes. Detecting inconsistency in the data, allows the Context-Manager to find possible broken sensors or, even more, avoid unpleasant situations where precise information is needed. The Context-Manager is mainly responsible for detecting and handling context changes that appear, and as a result, taking the proper actions. Another important task for the Context-Manager is to acquire user input and to decide if it is relevant or not. It is composed two elements: the Awareness Engine and the Adaptation Engine.

1) The Awareness Engine

It is the main component of the Proactive Filter. It is managing the data coming from sensors, which are in charge of detecting possible context changes. For smartphones, physical sensors are providing important information about the user's location, motion and mobility. Also, other information that comes from logical and virtual sensors like the user's interests, activities and set of used applications is constantly analyzed. Accessing this kind of information should be limited to some extend and controlled as it represents a privacy issue.

2) The Adaptation Engine

This component is crucial, as it is used for dealing with *unexpected events* and for ensuring that adaptive actions are

performed in a smooth cooperation between the main sub-parts of the Proactive Engine. Also, it has to check the constraints and the conditions of the system before adaptation and if the system will still behave according to its policies. For example, adaptation could involve the interface of an application that can be modified according to the needs of the user. If a user is on the move and the application detected a speed higher than 10 km/h while the user is using the application, the interface would be change to contain a layout with bigger button, bigger writing and brighter colors.

B. The Rules Engine

The Rules Engine is responsible for maintaining a precise overview of the system's goals and for running Proactive Rules. It keeps a list of required actions that would come as a response in case *expected events* occur. It is also used for storing the state of the system. Executing multiple Proactive Rules in parallel is due to its integrated Queue System and it is one of the great functionalities of the Rules Engine. The Rules Engine executes Proactive Rules in *Iterations*. A constant time interval is set between two consecutive *Iteration*s. It has a default value, but it can be adjusted depending on the performance of the device that is running the MPE. For example, initially, the time interval is set to 5 seconds, as there are cases where the MPE needs to check for events in a periodic manner. If at one Iteration, the Rules Engines has to execute a big number of Proactive rules, the time interval can be modified accordingly in order to allow resources to be efficiently shared between multiple threads and processes that are running on the MPE.

Proactive Rules can be used for serving multiple purposes: for checking context situations, for detecting special events, for analyzing contextual information, for synchronizing sub-parts of the model, for saving useful data into the Local Database, for sending rules and commands to other PE and for sending content to the Notification Manager. The Awareness Engine and the Adaptation Engine also posses the ability of activating Proactive Rules as well as the other way around. Another important property of WProactive Rules is that they can run at each Iteration of the Rules Engine or, in case they perform only simple actions, they can run only once and then finish their execution.

C. The Notification Manager

The purpose of the Notification Manager is to deliver informative content to the user. The content can take various forms like hints, messages, notifications or alarm. This is a crucial part of the entire model as it helps in achieving his/her goals, guides him/her in multiple situations and informs the user about certain events. The Notification Manager is in a close cooperation with the device's Operating System for handling messages. Proactive Rules prepare the content of the message, which is then forwarder to the user through the Notification Manager. For mobile phones, notifications can appear on the screen for short periods of time, messages can be registered by the Operating System and can be read later on by the user, hints are displayed as short text boxed for guiding the user when he/she is interacting with different applications and alarm, which can also be set by Proactive Rules, are registered in the integrated calendar and triggered for announcing the user in case of an upcoming important event.

V. NETWORKS OF MOBILE PROACTIVE ENGINES

MPEs are designed to work both offline and online. Having a network of distributed MPEs that communicate and exchange data provides a great opportunity for these systems to gain useful information. This way, MPEs are not only gathering data from their internal sensors but also from other MPEs. By design, information sharing between devices using MPEs is conceived to be done in a transparent way, without the implicit command of the user. Only in case of special situations and because of privacy issues, where sensible data is involved, the users should be asked to take over and decide what would be the next action the MPE should take. The most significant aspect to be taken into consideration is the actual information that is gained by a MPE when it gets data from other MPEs. One case is to find common interest or preferences between users that are working with applications having an integrated MPE. For example, a user could be looking for a ride on a car-sharing web site. Another user, which would be located nearby, maybe from the same city, would be looking for a ride having the same destination and exactly on the same dates. The MPEs would notify both users and would propose to share a ride for reducing the costs. Another case where data exchanging is useful is when a MPE is not sure what action to take and how to adapt its behavior when *unexpected events* happen. Requesting feedback from other MPEs that have more information is a possible solution for taking the right decision.

If we take, for example, two MPEs, one that was offline for a long period of time and one that was online during the

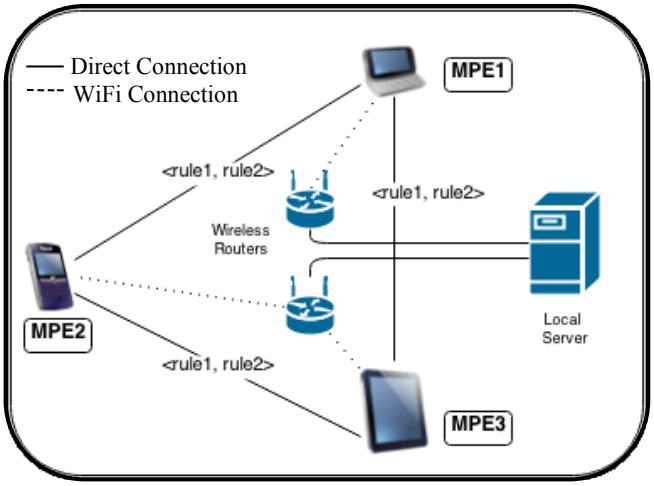

Figure 4. A possible network of distributed MPEs

same period of time. And now, both of the MPE should be able to share information because they would be having access to a communication channel between them. The MPE that was offline could learn a lot from the online MPE that stored information about its previous tasks and about the older state of the system, without using the Adaptive Engine, the Awareness Engine and the Rules Engine to process similar data and to go through the same adaptation process. As a consequence, local resources and time could be saved.

A special aspect to taken into consideration is the interaction of the users with their MPEs. Their behavior can be analyzed by the MPEs and stored for future decisions. If a MPE would detect, for instance, that its user is changing the mode of the phone each morning on the first day of the week, due to certain activities, it could start performing this action automatically on behalf of the user. The user is important as it still remains in the processing loop, but only for supervising purposes and for providing input in cases where information cannot be acquired from different sources. Sometimes access to personal information like the location of the user, his/her local preferences, his/her status, the applications that he/she is using or his/her list of contacts should be only granted with the special consent of the user.

Fig. 4 shows a possible scenario of a network of distributed MPEs. Three devices, with a running MPE, located on the same LAN, are connected to the Internet through a WiFi connection. A direct connection can be also established via Bluetooth, via Near Field Communication (NFC) techniques or via Android's WiFiP2P library for smartphones. The advantage of having a direct connection between the devices, illustrated in Fig. 3 with a straight line, is that Proactive Rules are exchanged immediately, without having to be sent firstly to a server. This means that each device equipped with a MPE will be acting like a server, being able to receive and send data to other devices having an integrated MPE.

Different network topologies can be employed for creating groups of MPEs: centralized, hybrid and peer-to-peer. The centralized topology means that a server can be used to handle the connections and the communication of MPEs. This is the easiest way to ensure their communication but it has a major flaw, a single failure point, the server. In order to address this issue, a hybrid topology can be used, where multiple servers are available over the network. However, this method increases the complexity of the communication between MPEs as they have to reason about the synchronization part between the servers. Complex algorithms should then be used to be sure that all the MPEs got the necessary information in case of collaborative actions. The complexity and workload distribution rapidly increase when the number of MPEs increases. In case of peer-to-peer networks, MPEs would be able to exchange information directly. This can happen however when the devices are close to each other and can send are receive bigger amount of data. For example, in a smart home, MPEs could be arranged into a peer-to-peer network. The devices could stay connected and be aware of all the other devices in their network.

VI. Case Study: An E-Learning Mobile Application

To better illustrate the behavior of a MPE and the usefulness of having a network of MPEs, we created an example of a possible scenario for its practical implementation. For simplicity, we focused more on describing the possible situations that highlight the benefits of having a network of MPEs and not on the implementation details. All around the world, students are using online e-learning platforms, like Moodle™ [37], for accessing educational content, completing assignments and participating in discussion related to their courses. These e-learning platforms are quite static as they are waiting for instructions or commands from their users. This is why an e-learning application for mobile devices, i.e., smartphones and tables, with an integrated Proactive Engine, would come in hand. We assume that the application would be directly connected with the web platform and would have access to all the data from the student's account on the LMS.

The application would include an advanced mechanism for displaying notifications and questions for the user, provide hints and trigger alarms. Hints would be used for guiding the user, questions for asking for specific instructions, notifications as short messages to inform the user and alarm to alert him/her in case of extraordinary situations or events. On one hand, even though these features have been integrated in other E-Learning applications, the way they are created and handled represents the novelty in this case. On the other hand, they are already addressing some of the major issues when using an online e-learning platform. These issues appear because of the lack of an immediate notification channel between the students or between the students and the professors in case extraordinary situations appear. Certain online platform have an online mechanism for enrolling to an exam, and students often miss these deadlines, resulting in a big problem both for the student and the administration of universities and schools. More issues include missing deadlines for assignments and nonparticipating in forums.

For instance, the scenario of an instructor that has to give an exam on a specific date, at a specific hour and would be late due to traffic is an example of a foreseen event. The logical sensors of the MPE would know that there is an exam approaching soon based on the calendar of the LMS, where the exact date and hour of the exam would be set. The MPE would send alerts to the instructor and he/she could post a short message, via his/her smartphone, on the forum of the course announcing that he/she will be late. Not only will the students be notified of this, but a person from the administration could also alert the students in person if they would not have their device with them. The physical sensors of the MPE would sense that he is moving and so, would adjust the graphical user interface for writing messages.

```
Proactive Rule R001
Description: This Rule is designed to run on each MPE in order to check for new connections in the same network with which the current MPE could share information if they are working on the same assignment.

data acquisition
    conn [] = getConnectionsOnSameNetwork()
activation guards
    conn.size != 0
conditions
    conn.assignment.isStillValid()
actions
    foreach connection in conn []
      if(usersWorkOnSameAssignment(
      connection.assignemnt.ID))
        sendMessageToMPE(conn.ID, message)
        inviteOtherMPEforCollaborativeWork(
        connection.assignemnt.ID)
      end if
    end foreach
rules generation
    if(!activationGuard)
      createRule002(conn.ID, conn.assignment.ID)
    end if
    cloneRule (R001)
```

Figure 5. An example, in pseudo-code, of a Proactive Rule

```
Proactive Rule R002
Description: This Rule is designed to constantly check if the was any update in the deadline or additional documents were added for this assignment.

data acquisition
    connID = getConnectionID()
    assigID = getAssignmentID()
activation guards
    return deadlineIsStillValid(assigID)
conditions
    return contentWasAdded(assigID)
actions
    if( conditions())
        extractAdditionalContent(assigID)
        alertUsersAboutAdditionalContent (users[])
        updateAlarmOfAssignment(assigID)
    end if
rules generation
    if(!activationGuard)
      cloneRule (R002)
    end if
```

Figure 6. A second Proactive Rule in pseudo-code, generated by the first Proactive Rule R001

More advance actions would include setting an alarm for deadlines, putting the events into an integrated calendar, proposing to students to collaborate on solving assignments with other classmates, which are close to their location or , even more, automatically download documents or course material directly to the smartphones of the students. The majority of these actions are not currently provided by any existing LMS and, adding plugins or third party applications will not change the overall behavior of the system. These actions represent collaborative actions between the LMS and the clients. However, MPEs allow another type of collaboration, where each MPE is part of the process. In addition to traditional forms of communication between mobile devices like calls and messages, MPEs are capable of exchanging relevant context information, checking if there are any errors or mistakes on the data acquired from their local sensors, reaching for remote data, sharing resources and of allowing synchronous/ asynchronous communication. This type of innovative collaboration allows MPEs to provide sophisticated real-time services and to support complex mobile applications.

In Fig. 5, an example of a Proactive Rule, which would be used for this case study, is illustrated in pseudo-code. More specifically, this Proactive Rule would run at each iteration of the Rules Engine and would get activated only when there would be at least two MPEs on the same network. Its purpose is to invite the users of the MPEs, in case they are working on the same assignment, to collaborate and share their knowledge. The *proactive* aspect comes from the fact that this situation is anticipated by the MPEs, without any specific intervention or command from the users of the MPEs.

First, Proactive Rules would check for constraints on the LMS to see if the assignment was made as a collaborative assignment or as an individual assignment. In case the assignment is open to cooperation between students, the Awareness Engine would be alerted. It would then be in charge of two main tasks: detecting from the internal virtual sensors of the MPE that a user is actively working on the assignment and discovering other MPEs that are available on the same network and open to collaboration. Then, Proactive Rules, like the one illustrated in Fig. 5, would get activated and would start analyzing and deciding if there were any real-time possibility of collaboration between the two MPEs. If yes, the Adaptation Engine would trigger different actions, like messages to the users to ask them if they want to collaborate remotely or meet and solve the assignment in case the MPE would detect that they are close to each other, e.g., both MPEs would be connected to the same Wi-Fi network in the campus.

In Fig. 6, a second example of a Proactive Rules is given to illustrate that several Proactive Rules can be generated and executed at the same time by a MPE. Rule R002 was activated by the rule R001 for checking, at every iteration of the Rules Engine, if there were any updates in the deadline of assignment or if additional learning material was added on the LMS. In case these events occur, the users are notified immediately via email or via messages sent trough the Notification Manager and, the alarm that was automatically

set by the MPE will be updated in order to announce the users in advanced about their deadline. Rule R002 will not activate any other Proactive Rule, it will just clone itself until the deadline of the assignment has been reached.

VII. OTHER FIELDS OF APPLICATIONS FOR MPES

The previous case study indicated that MPEs could be used in education, in a computer supported collaborative work scenario. MPEs could also be implemented into application in other domains, which could benefit from the possibility of having multiple networks of systems capable of performing collaborative actions, like medicine, public transportation, tourism, business and many social networks platforms.

In hospitals, for instance, MPEs could be integrated into various medical systems with computing capabilities that would communicate for providing better services to the patients. All these interconnected devices would have access to a variety of sensors, which could lead to a better anticipation of emergencies and a better response from the medical staff. In transportation, for instance, MPEs could help preventing traffic jam, based on the experiences of other MPEs. If the system would detect that multiple devices are moving very slowly on a section where the high speed is allowed it could alert the other MPEs about the situation and an alternative route would be proposed. In tourism, users could experience new ways of connecting with the surrounding environment though the use of MPEs that could obtain relevant information based on their location.

VIII. MPE IMPLEMENTATION

The architecture proposed for mobile devices in this paper is currently under development for the iOS-based devices. A basic working prototype application for smartphones and tablets running an Android Operating System has been already created and is already being tested with basic sets of Proactive Rules. The Rules Engine was implemented using Java and it is capable of running Proactive Rules. The local database and the connection between the Rules Engine and the database were implemented using SQLite™ [38] and ORMLite™ [39]. The MPE is able to communicate with other MPEs via Google's GCM framework [40] that allows devices to exchange messages over the same connection. The Adaptation Engine and the Awareness Engine will run on single background threads that are capable of monitoring the environment, capture data from various sensors that are integrated into mobile devices and performing adaptation at the level of the interface. The Notification Manager is using the notifications built-in libraries of the Android Operating System. The graphical user interface is still subject to modifications, as it depends on the specific requirements to which the application will serve. A complete description of its implementation and of its performance will soon follow, after the prototype will be finalized.

IX. CONCLUSION AND FUTURE WORK

In this paper, we have outlined that we are heading towards various distributed networks of proactive, context-aware and self-adaptive systems. Our reasoning is supported by the continuous intensive research initiatives of many laboratories in the fields of Context-Awareness, Proactive Computing and Self-Adaptation.

Our work opens new substantial research possibilities and new perspectives on how future smart applications will behave, communicate and collaborate. To support our vision, we proposed a model for mobile devices that is able to integrate all the discussed properties. Designers can now focus more on high-level implementation planning and on the functionalities that their applications will provide, than on architectural design, configuration details and on compatibility issues.

Adaptability is provided at a behavioral level, using the model of feedback loops, and at the level of the communication between the different components of our model. Awareness comes from the fact of taking into account different contexts and Proactivity is achieved by using a rules-based engine for handling foreseen events. With the proposed model, the user is focusing more on how to interact with the application and not how to manage and configure the system.

A. Challenges Ahead

Two of the most challenging points in the close future are to develop techniques for communication and collaboration between MPEs and to design and develop smart applications based on the proposed model taking in account important factors like user mobility, different computing capabilities of various devices and privacy issues.

B. Future work

A case-study based evaluation will follow for validating all the characteristics of the presented model and for answering to some research questions such as whether or not the model is correctly providing routines in a context-adaptive manner, or if the parts of the model are really taking into account the user's preferences, or if the model has self-adaptive properties that allows it to modify its behavior. In the upcoming case study, the application presented as the motivating scenario in this paper will be implemented for mobile devices. Proactive Rules will be developed to take care automatically of the tasks that can be performed by the application without asking for specific commands from the user.

Important research is still to be done for exploring the computing capabilities of other intelligent devices that could support proactive context-aware adaptive applications, not only mobile devices. Networks of wearable devices, ubiquitous devices and other computing systems could be thus connected to form a global distributed intelligent network.

REFERENCES

[1] R. A. Dobrican and D. Zampunieris, "Moving Towards a Distributed Network of Proactive, Self-Adaptive and Context-Aware Systems, " in Proc. of the 6th International Conference on Adaptive and Self-Adaptive Systems and Applications, (ADAPTIVE), May 2014, pp. 22-26.

[2] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Transactions on Autonomous and Adaptive Systems, 2009, vol. 4, pp. 1-42.

[3] Y. Brun, G. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Muller, M. Pezze, and M. Shaw, "Engineering Self-Adaptive Systems through Feedback Loops," in Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, Springer, 2009, vol. 5525, pp. 48-70.

[4] S. Dobson, S. Denazis, A. Fernandez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," ACM Trans. Auton. Adapt. Syst., 2006, vol. 1, pp. 223-259.

[5] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. "Model evolution by run-time parameter adaptation, " in Proc. of the 31st International Conference on Software Engineering, ICSE 2009, IEEE, pp. 111-121.

[6] IBM Corporation: "An Architectural blueprint for autonomic computing," White paper, 4th edn., IBM Corporation, 2006.

[7] J. Indulska and P. Sutton, "Location management in pervasive systems," in Proceedings of workshop conference on the Australasian information security ACSW frontiers, vol. 21, January 2003, pp. 143-151.

[8] S. Kurkovsky, "Location-dependent and Context-Aware Computing," Next Generation Mobile Networks and Ubiquitous Computing, 2001, 156.

[9] B.H.C. Cheng, R. de Lemos, H. Giese, P. Inverardi,J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Muller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, Springer, 2009, vol. 5525, pp. 1-26.

[10] D. Tennenhouse, "Proactive Computing," Communications of the ACM, 2000, vol. 43, issue 5, pp. 43-50.

[11] A. Oulasvirta and A. Salovaara, "Six modes of proactive resource management: a user-centric typology for proactive behaviors," in Proc. NordiCHI 2004, ACM Press, pp. 57-60.

[12] D. Zampunieris, "Implementation of a Proactive Learning Management System," in Proc. E-learn 2006, AACE Press, pp. 3145-3151.

[13] S. Coronado and D. Zampunieris, "Towards a proactive learning management system using early activity detection," in SITE08, AACE Publishing, 2008, vol. 1, pp. 306-311.

[14] R. Dobrican, S. Reis, and D. Zampunieris, "Empirical Investigations on Community Building and Collaborative Work inside a LMS using Proactive Computing," in Proc. E-learn 2013, vol. 1, pp. 1840-1852.

[15] R. Dobrican and D. Zampunieris, "Supporting collaborative learning inside communities of practive through proactive computing," in Proc. EDULEARN13, 2013, pp. 5824-5833.

[16] D. Shirnin, S. Reis, and D. Zampunieris, "Experimentation of Proactive Computing in Context Aware Systems: Case Study of Human-Computer Interactions in e-Learning Environment," IEEE CogSIMA, Feb. 2013, pp. 269-276.

[17] V. C. Ostuni, T. Di Noia, R. Mirizzi, D. Romito, and E. Di Sciascio, "Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia," in SeRSy, October 2012, pp. 37-48.

[18] P. Coppola, V. Della Mea, L. Di Gaspero, S. Mizzaro, I. Scagnetto, A. Selva, L. Vassena, and P. Riziò, "MoBe: context-aware mobile applications on mobile devices for mobile users," in Proc. of the International Workshop on Exploiting Context Histories in Smart Environments, 2005, Munich, Germany, pp. 8-13.

[19] E. Bertou and S. Suleman, "OnRoute: A Mobile Context-Aware Public Transportation Planning Application," in HCI International 2013-Posters' Extended Abstracts, Springer Berlin Heidelberg, 2013, pp. 299-303.

[20] J. Yim and C. Le. Thanh, "Museum Guide, a Mobile App," in Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity, Springer Berlin Heidelberg, 2012, pp. 36-41.

[21] D.B. Abeywickrama, N. Bicocchi, and F. Zambonelli, "SOTA: Towards a General Model for Self-Adaptive Systems," 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, June 2012, pp. 48-53.

[22] J. Hakkila, and J. Mantyjarvi, "Collaboration in context-aware mobile phone applications," in Proc. of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS'05, IEEE, 2005, pp. 33a-33a.

[23] B.H.C. Cheng, R. de Lemos, P. Inverardi, and J. Magee (Eds), "Software engineering for self-adaptive systems," in Dagstuhl Seminar, 2009, vol. 5525.

[24] W. Heaven, D. Sykes, J. Magee, and J. Kramer, "A case study in goal-driven architectural adaptation," in Software Engineering for Self-Adaptive Systems, 2009, Springer Berlin Heidelberg, pp. 109-127.

[25] A. A. Mansor, W. M. W. Kadir, and H. Elias, "Policy-based approach for dynamic architectural adaptation: A case study on location-based system," in 5th Malaysian Conference on Software Engineering (MySEC), IEEE, December 2011, pp. 171-176.

[26] M. M. Islam, M. A. Sattar, M. F. Amin, X. Yao, and K. Murase, "A new constructive algorithm for architectural and functional adaptation of artificial neural networks," in Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 6, 2009, pp. 1590-1605.

[27] Z. Yang and Z. Jin, "Modeling and Specifying Parametric Adaptation Mechanism for Self-Adaptive Systems," in Requirements Engineering, Springer Berlin Heidelberg, 2014, pp. 105-119.

[28] G. Tamura, N. M. Villegas, H. A. Müller, L. Duchien, and L. Seinturier, "Improving context-awareness in self-adaptation using the DYNAMICO reference model," in Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press, 2013, pp. 153-162.

[29] H. Klus, D. Niebuhr, and A. Rausch, "A component model for dynamic adaptive systems," in International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting, ACM, 2007, pp. 21-28.

[30] R. Mizouni, M. A. Serhani, A. Benharref, and O. Al-Abassi, "Towards Battery-Aware Self-Adaptive Mobile Applications," in Proc. of the 9th International Conference on Services Computing (SCC), IEEE , June 2012, pp. 439-445.

[31] A. Karadimce and D. C. Bogatinoska, "Using hybrid mobile applications for adaptive multimedia content delivery," in Proc. of the 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 2014, pp. 686-691.

[32] T. Ruiz-López, C. Rodríguez-Domínguez, M. J. Rodríguez, S. F. Ochoa, and J. L. Garrido, "Context-Aware Self-adaptations: From Requirements Specification to Code Generation," in Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction, Springer International Publishing, 2013, pp. 46-53.

[33] M. Hussein, J. Han, and A. Colman, "An Approach to Model-Based Development of Context-Aware Adaptive Systems," in Proc. of the 35th Annual Computer Software and Applications Conference (COMPSAC), July 2011, IEEE, pp. 205-214.

[34] L. David, M. Endler, S.D.J. Barbosa, and J.V. Filho, "Middleware Support for Context-Aware Mobile Applications with Adaptive Multimodal User Interfaces," in Proc. of the 4th International Conference on Ubi-Media Computing (U-Media), July 2011, pp. 106-111.

[35] M.T. Segarra and F. Andre, "Building a Context-Aware Ambient Assisted Living Application Using a Self-Adaptive Distributed Model," in Proc. of the 5th International Conference on Autonomic and Autonomous Systems, ICAS '09, April 2009, pp. 40-44.

[36] J. Yau and M. Joy, "A Context-aware and Adaptive Learning Schedule framework for supporting learners' daily routines," in Proc. of the 2nd International Conference on Systems, ICONS '07, April 2007, pp. 31-37.

[37] Moodle - Modular Object-Oriented Dynamic Learning Environment. [retrieved: April, 2014]. Available from: https://moodle.org/

[38] SQLite Framework. [retrieved: April, 2014]. Available from: http://www.sqlite.org/

[39] ORMLite - Lightweight Object Relational Mapping (ORM) Java Package. [retrieved: April, 2014]. Available from: http:// http://ormlite.com/

[40] Google GCM – Google Cloud Messaging for Android. [retrieved: August, 2014]. Available at http://developer.android.com/google/gcm/index.html