Moving Towards a Distributed Network of Proactive, Self-Adaptive and Context-Aware Systems

Remus-Alexandru Dobrican, Denis Zampunieris
Computer Science and Communication Research Unit, University of Luxembourg
Luxembourg, 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 behavior 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 engineering challenge. Even though there have been notable steps towards distributed self-adaptive and context-aware systems, there is still a lack of methodologies on how to model and implement applications which have to distribute and to manage large amounts of information. In this work-in-progress, we address this issue by proposing a self-adaptive and context-aware model with a structure that allows the system to learn from the user's behavior by using Proactive Computing. The novelty comes from the possibility of having a distributed network of Proactive Engines in which the exchange of contextual information would help each system to take smart decisions.**

Keywords—self-adaptive systems; context-aware systems; proactive computing; distributed network.

I. INTRODUCTION

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 [1]. Giving instructions to complex software systems is becoming quite a difficult task for 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.

Self-adaptation in software systems comes in many different aspects. Self-adaptive systems can be characterized by their operating mode which easily permits them to fulfill their goals in a modified context. Feedback loops provide an architectural solution for self-adaptation. Brun et al. [2] indicate that feedback loops usually include four key activities: *collecting*, *analyzing*, *deciding* and *acting*. These activities are essential for achieving self-adaptability. In Figure 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.

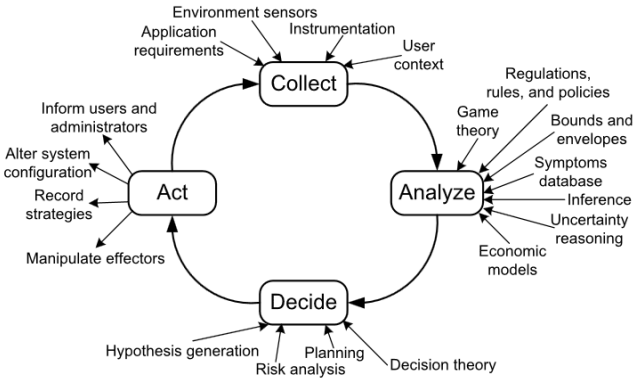

Figure 1. Autonomic control loop [3]

Context-aware systems are designed to continuously analyzing contextual information, which is a key feature for determining the occurrence or the lack of events.

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*) [4].

Tennenhouse [5] firstly 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 [6], 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-adaptiveness handle unforeseen events, which are seen as deviations from the normal situations.

The contribution of this paper is two-fold. First, it offers an infrastructure for software systems capable of performing automated tasks for the user, of analyzing large quantities of data and making decision in different contexts. Second, it provides an analysis of a distributed network of systems that are implementing our model.

The rest of the paper is organized as follows: Section 2 describes the main characteristics of a Proactive Engine. Section 3 investigates the possibility of having a distributed network of Proactive Engines. Section 4 provides an example of application for our model; other applications are proposed. Section 5 conclusions about the potential of Proactive Engines.

 ISBN: 978-1-61208-341-4

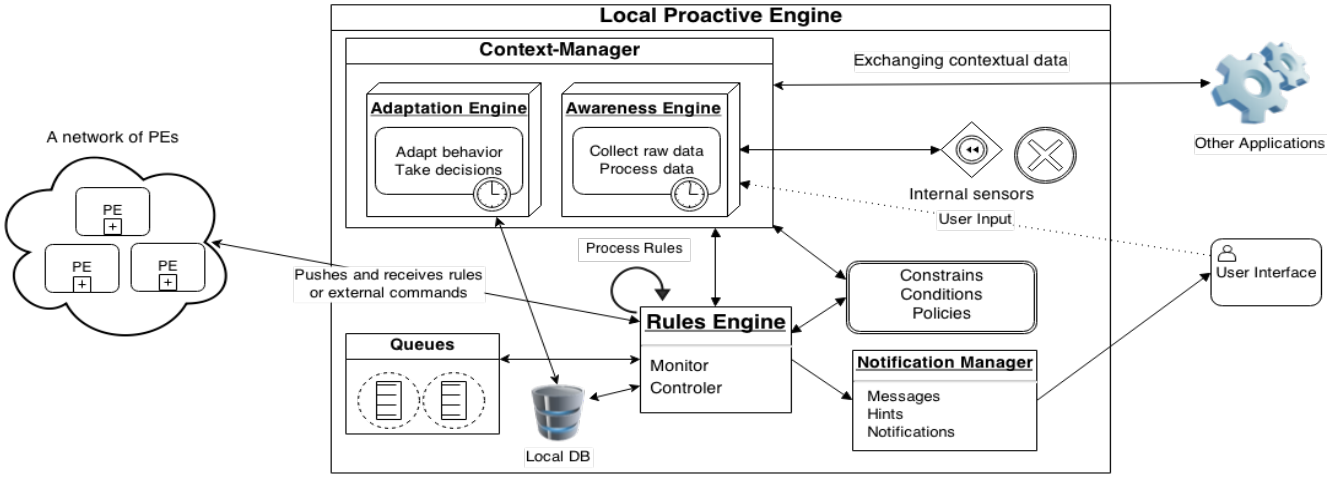

Figure 2. The infrastructure of a Proactive Engine

II. PREVIOUS WORK

Zampunieris developed the concept and the structure of the first Proactive Engine in 2006 [7]. It 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) [8]. 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 [9]. Previous work, [10] and [11], focused until now on applying Proactive Computing on a single system, thus exploring only the scenario of having only one centralized Proactive Engine. 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. The possibility of having an entire network of Proactive Engines exchanging data and learning from each other was not yet explored.

III. PROACTIVE ENGINES

We propose a new version of the Proactive Engine, 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; now, each step is assigned to a specific structure. 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 LPE 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 LPE 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 Figure 2).

A. The Context-Manager

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

This component is managing the data coming from sensors, which are in charge of detecting possible context changes. For smartphones, sensors are providing important information about the user's location, motion and preferences. For PCs, the information would focus more on the user's interests, activities and set of used applications. 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.

 ISBN: 978-1-61208-341-4

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 an *expected event* shows up. 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. 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 have the ability to activate Proactive Rules.

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.

IV. A NETWORK OF PROACTIVE ENGINES

PEs are designed to work both offline and online. Having a network of distributed LPEs that communicate and exchange data provides a great opportunity for these systems to gain useful information. This way, LPEs are not only gathering data from their internal sensors but also from other LPEs. By design, information sharing between devices using LPEs is conceived to be done in a transparent way, without the implicit command of the user.

Figure 3 shows a possible scenario of a network of distributed LPEs. Three devices, with a running LPE, 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 figure 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 with a LPE will be acting like a server, being able to receive and send data to other devices with LPEs.

The most significant aspect to be taken into consideration is the actual information that is gained by a LPE when it gets data from other LPEs. One case is to find common interest or preferences between users that are working with applications having an integrated LPE. 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 LPEs would notify both users and would propose to share a ride for

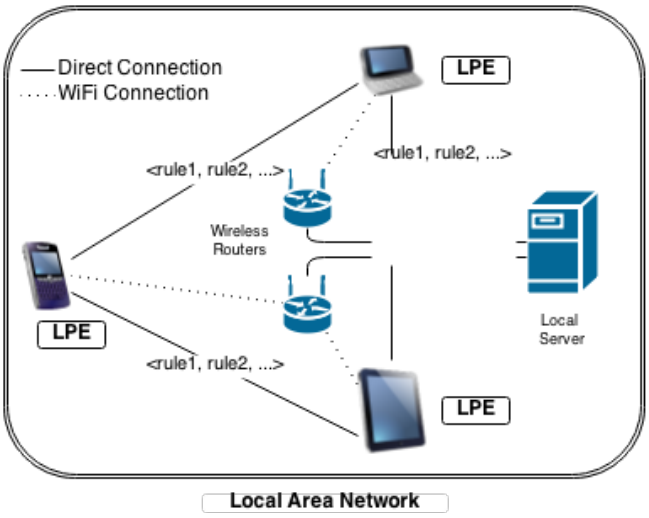

Figure 3. A possible network of distributed LPEs

reducing the costs. Another case where data exchanging is useful is when a LPE is not sure what action to take and how to adapt its behavior when *unexpected events* are appearing. Requesting feedback from other LPEs that have more information is a possible solution for taking the right decision.

If we take, for example, two LPEs, one which was offline for a long period of time and one which was online during the same period of time. And now, both of the LPEs would be able to share information because they would be have access to a communication channel between them. The LPE that was offline could learn a lot from the online LPE 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.

V. CASE STUDY

To better illustrate the behavior of a LPE and the usefulness of having a network of LPEs, 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 LPEs and not on the implementation details.

All around the world, students are using online e-learning platforms, like Moodle™ [12], 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 PCs and for smartphones, 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.

 ISBN: 978-1-61208-341-4

The application would include basic actions like 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/and events.

Even though these actions are quite elementary, 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 example, if an instructor were to give an exam on a specific date, at a specific hour, and is late due to traffic, 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 sensors of the LPE would sense that he is moving and so would adjust the graphical user interface for writing messages.

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 private PCs or 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.

In Figure 4, an example of a Proactive Rule, which would be used for this case study, is illustrated in pseudo-code. More specifically, it is a Global Meta-Scenario because it runs at each iteration of the Proactive Engine and because it is used only when there are at least two LPEs on the same network. Its purpose is to invite the users of the LPEs, 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 Global Meta-Scenario, without any specific intervention or command from the users of the LPEs.

There are five main parts that compose a Proactive Rule: data acquisition, activation guards, conditions, actions and rules generation. The first part is used for gathering useful data, in this case if there are new connections or LPEs available on the same network, the second and the third part are used for checking for special conditions and constraints, like if the users of the LPEs have common assignments, and the fourth and the fifth parts are used to take specific actions, like sending a personalized messages to the users of the LPEs, and to generate other Proactive Rules.

```
Global Meta-Scenario (GMS) 001
Description: This Rule is designed to run on each
LPE in order to check for new connections in the
same network with which the current LPE 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))
        sendMessageToLPE(conn.ID, message)
        inviteOtherLPEforCollaborativeWork(
        connection.assignemnt.ID)
      end if
    end foreach
rules generation
    if(!activationGuard)
      createGMS002(conn.ID, conn.assignemnt.ID)
    end if
    cloneRule (GMS 001)
```

Figure 4. An example, in pseudo-code, of a Proactive Rule

A. Other fields of applications for LPEs

The previous case study indicated that LPEs could be used in education. In hospitals for example, LPEs could share information and create very accurate and useful reports for doctors. They could also be implemented in other domains, which have to handle big amounts of data coming from sensors, like other areas of medicine, transportation, engineering, aviation and many social networks platforms.

VI. A Short Implementation Overview

We are currently working on the implementation of LPEs for smartphones and tablets, with an Android Operating System, as they allow direct data exchange between devices which are on the same Wi-Fi network, without having an intermediate access point. Frameworks like Android's WiFi-P2P, SQLite™ [13] and ORMLite™ [14] will be used for creating the prototype. The Proactive Engine will run as a background service.

VII. Conclusion and Future Work

In this work-in-progress, we have identified and described a few of the important features and characteristics of a model that achieves to integrate proactive, self-adaptive and context-aware features into software systems. With the proposed model, the user is focusing more on how to interact

 ISBN: 978-1-61208-341-4

with the application and not how to manage and configure the system.

A. Challenges Ahead

Two of the most challenging points are to ensure the communication between the components of a LPE and to design proactive scenarios, while 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 allow it to modify its behavior.

REFERENCES

[1] 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.

[2] Y. Brun et al., "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. 48-70.

[3] S. Dobson et al., "A survey of autonomic communications". ACM Trans. Auton. Adapt. Syst., 2006, vol. 1, pp. 223-259.

[4] B.H.C. Cheng et al., "Engineering Self-Adaptive Systems through Feedback Loops" in Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, Springer, 2009, vol. 5525, pp. 1-26.

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

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

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

 ISBN: 978-1-61208-341-4