Management of Mobile Dynamic Adaptation in Cyber-Physical Systems

Rafael Oliveira Vasconcelos, Igor Vasconcelos, Markus Endler
Department of Informatics
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
{rvasconcelos,ivasconcelos,endler}@inf.puc-rio.br

***Abstract*—The general idea of CPS (Cyber-Physical Systems) is integrating information systems with the physical world. Thus, CPS require a dynamic global network infrastructure where devices are enabled to autonomously adapt to the cyber and physical world events. Conversely, devices frequently are mobile devices that can be moved or can move by themselves, such as sensors and actuators. Current solutions for building CPS do not provide the required support to handle changes in the cyber/physical context and in particular, deal with the devices' mobility. In this paper, we present an approach that supports distributed dynamic software adaptations among mobile devices.**

Keywords—mobile dynamic adaptation; mobile communication; middleware; adaptability; SDDL

I. INTRODUCTION

A CPS may be defined as a networked information system (i.e., cyber world) that is coupled with the physical world through a huge number of geographically distributed devices (e.g., vehicles and smartphones), sensors (*e.g.*, temperature, speed) and actuators (*e.g.*, electro-mechanical, electronic or other controllable devices)[1]. The authors [2] explain that CPS should have mechanisms and services to make them aware of changes in the cyber and physical contexts with the aim of adapting the system´s execution according to these contexts. Reasons for adaptation include hardware defects, software errors, sudden resource changes, or bursts of communication or processing demand. On the other hand, many CPS must keep executing while the system is dynamically adapted [3].

In many applications, the CPS devices are mobile ones, such as smartphones, wearable devices, vehicles or autonomous robots with specific sensors and actuators. A typical scenario is thus the discovery and *ad hoc* interaction among such mobile devices (a.k.a. mobile nodes – MNs) and for mutual exchange of sensed data. For example, when a user carrying a smartphone enters a vehicle, the mobile application on the phone could connect and interact with the vehicle´s control system in order to provide some features, such as local pre-processing and transmission of the vehicle telemetric data or in-vehicle adjustment of equipment and media (e.g. seat and mirror position, music selection, etc.), according to user preferences. However, there are many challenges to build such applications since vehicles have different types of sensors, of actuators, general capabilities, and APIs (Application Programming Interfaces). Therefore, it is not feasible to develop from scratch a mobile application for all possible vehicles and passenger customization options. The two main reasons are: (i) time and cost of developing the integration mechanisms/protocols among the application and the different vehicles, and (ii) limitations on the allowable application size. The best possible approach is to implement the integration software for each vehicle type or model as a component (or plug-in). This allows one to independently add the specific software component for new sensors and actuators of new vehicle models as needed, and even do this dynamically during the application´s execution.

Current solutions for building adaptive mobile systems do not provide the required support to handle dynamic changes in the cyber/physical context of the devices [2]. The authors of [2] also claim that there are few efforts on how to adapt and tailor existing software to the specific aspects of CPS (e.g., high dynamicity and distribution, real-time reactivity, resource constraints, and error/defect-prone environments). One should also add scalability, fault-tolerance, mobility and dynamic adaptation as further important issues when designing software for a "*Mobile CPS*". To the best of our knowledge, there are no works that cope with all the aforementioned aspects. As the main goal of our research is to facilitate the development of adaptive software for Mobile CPS, we present an approach that supports distributed dynamic adaptation among MNs that encompass a CPS.

The remainder of this paper is organized as follows. Section II provides an overview of the key concepts and technologies used in our approach. Section III presents our proposed approach for mobile dynamic adaptation in Mobile CPS. Section IV enlightens the proposed evaluation. Section V reviews the most relevant works related to ours. Finally, Section VI contains concluding remarks and discusses future work on the central ideas presented in this paper.

II. BASIC CONCEPTS

This section presents and discusses the main concepts and technologies that underlie our approach of dynamic adaptation.

A. Scalable Data Distribution Layer (SDDL)

SDDL [4] is a communication middleware that connects stationary DDS (Data Distribution Service) [5] nodes in wired "core" network to MNs with an IP-based wireless data connection. SDDL employs two communication protocols: Real-Time Publish-Subscribe RTPS Wire Protocol [5] for the wired communication within the SDDL core network, and the Mobile Reliable UDP protocol (MR-UDP) [6] for the inbound and outbound communication between the core network and the mobile nodes. DDS delivers many advantages in performance, scalability, availability and QoS (Quality Of Service) mechanisms [7] [8].SDDL provides a communication infrastructure to interconnect sensor, actuators and other mobile devices so to build CPS systems. Thus, we have chosen build our work based on the SDDL.

As part of the SDDL core, the Gateway plays an important role concerning our work. It defines a unique point of attachment for connections with the mobile nodes. Thus, the Gateway is responsible for managing a separate MR-UDP connection with each of these nodes, forwarding any application-specific message or context information into the core network, and in the opposite direction, converting DDS messages to MR-UDP messages and delivering them reliably to the corresponding Mobile Node(s).

B. Dynamic Software Adaptation

Dynamic software adaptation does behavioral, functional or structural modification of a software component (e.g., class, service, module or functionality) at run-time [9]. It allows application to act in response to new application requirements and/or context changes (e.g., in the physical or virtual world, [2] [3]. The two most popular approaches found in the literature for software adaptation are parameter and compositional adaptations [9].

Parameter adaptation is related to the change of software parameters (i.e., variables) with the aim of modifying the system behavior. However, with parameter adaptation, it is impossible to deploy new software components into a running system, and hence this type of adaptation is not sufficient for changing and extending a system's functionality. Conversely, compositional adaptation enables the exchange/addition of components in the system in order to satisfy new requirements and context changes that may arise after the software deployment [3]. Thus, compositional adaptation deals with situations where unforeseen requirements and conditions are common [9].

III. System Architecture

Our approach, illustrated in Fig. 1, supports parameter and compositional adaptation. One of the most widespread approaches in software engineering is to separate the application's business logic from adaptation logic [10] [11] [9] [2]. Thus, inspired by other works [10] [2], we provide an API where adaptation engineers can register for, and handle events (e.g., new device available, component failure, and events defined by the adaptation engineer) so to build their own application adaptation policies.

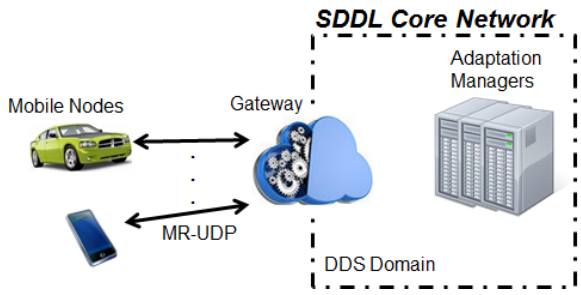

Fig. 1. Overview of the proposed approach

Adaptations performed at the MNs are driven/orchestrated from the SDDL Core Network. That is, the infrastructure is in charge of monitoring MNs, and then coordinating the execution of the adaptation by the MNs. Thus, the system nodes that compose our approach are *MN* and *Adaptation Manager*, which are explained in the next subsection.

A. System Nodes

The Adaptation Managers are responsible for coordinating (i.e., initiate and coordinate the execution of all the operations that encompass the decentralized adaptation) the system-wide adaption process (e.g. deployment of new software components or customization of application´s parameters) on many MNs. For example, if the global adaptation is the deployment of a new functionality to several MNs, which may consist of some components, the Adaption Manager will send the code that implements the new functionality to all MNs and then verify whether all MNs successfully deployed it.

Our approach further requires a Mobile Client Adaptation Service that executes at the MNs. This service executes the adaptation at the MN (e.g., deployment of a new functionality) and informs the Adaptation Manager whether this operation was successfully performed or not.

B. Management of Distributed Adaptation among Mobile Nodes

We have in mind that adaption is supposed to be atomic in some situations (i.e., all adaption is successfully performed or it has no effect). While some adaptations affect only a specific MN (i.e., there is no impact on other MNs), others may affect some (or all) MNs. To illustrate the latter case, we can imagine a situation where the adaptation engineer needs to update some component responsible for the interaction between MNs. In this case, different versions of such component may cause a flaw since MNs will communicate using different and incompatible versions of a component. Thus, the adaptation engineer has to inform that all, or none, MNs should perform the component update. We call this sort of adaptation as *transactional adaptation*, in contrast of *non-transactional* adaptation.

Typically, when a component has to be updated, it is blocked for new executions, the new component version is deployed and all references from the old version are updated to the new one. However, the time required to update the component may affect the system´s performance since the system will be blocked, if it tries to call such component, until

the update process finishes. To avoid such behavior, we propose the concept of *asynchronous* adaptation in which the old component does not need to be blocked while the new one is deployed. Thus, the system is able to operate normally while the component is updated. After the update process finishes, all references are changed from the old version to the new one. In some scenario, the adaption engineer may decide to asynchronously update the component in charge of compressing data sent by the MN to the infrastructure since the latter (i.e., SDDL Core Network) is able to decompress data using both versions and such behavior does not introduce inconsistence on the system.

A dynamic adaptation may cause some error or side effect (e.g., degrading the system´s performance). For such situations, we intend to support *adaptation rollback*. Hence, if some adaptation caused any undesirable situation, the adaptation engineer is able to restore the system to the previous system´s configuration.

IV. PRELIMINARY PERFORMANCE TESTS

So far, we have developed and tested only some features of our approach. More specifically, we implemented just non-transactional, synchronous and compositional adaptation into our current prototype. Hence, we are currently able to deploy new components and update existing components at the MNs. In our performance tests, we have measured the time required to deploy and to update a component (with non-transactional and synchronous adaptation), as well as the overhead of invoking (i.e., calling) a component through our component wrapper.

Our hardware test was composed of Dell Laptop Intel i5-3210M 2.5GHz, 8GB DDR3 1333MHz and Wi-Fi (802.11n) interface running Windows 8.1 64 bits, Dell Laptop Intel i5 M 480 2.66GHz, 8GB DDR3 1333MHz and Fast Ethernet interface running Ubuntu 12.04 LTS 64 bits, and a router with four Fast Ethernet ports and 802.11n wireless connection. Our prototype has been implemented using the Java programing language and we have simulated the MNs as a Java application.

In order to evaluate the time required to deploy or update a component, we did two experiments. One experiment measured the elapsed local time on the MN to complete the deployment/update of the component. In each case, we repeated the experiment 10 times. The JAR (Java ARchive) file that encapsulates the deployed component has 1.5KB (kilobytes) and the Java class that represents the component has 51 lines of code. While the first experiment measured a local time, the second experiment measured the Round-trip Delay (RTD), which encompasses the time interval from the instant of time the Adaption Manager sends a message until it receives an acknowledgment informing that the MN completed the deployment/update. We repeated the latter experiment with 1, 10 and 100 MNs. With the aim of assessing the invocation overhead that our wrapper imposes, we measured the time of 50,000 calls using direct invocation (i.e., calling the component's method without our wrapper) and invocation through the wrapper that represents the component.

The deployment of a component on the MN took 2.03 ms on average, while the process of update a single component instance took 0.05 ms on average. While the number of MNs was increased in a ratio of 100 times, the RTD increased less than two times for the deployment RTD and 2.3 times for the update RTD.

The time elapsed to execute 50,000 times the component´s method was 467.36 ms using the wrapper and 452.76ms calling directly the component´s method. Thus, the overhead imposed by our wrapper was 3.22%, which seems to be reasonable for the most of the applications considering the functionality provided by the wrapper.

V. RELATED WORK

Although not exhaustive, the works discussed in this section are the most relevant ones we encountered with respect to our approach. The work by [13] proposes a mobile application – Mobile Sensor Hub (MoSHub) – that allows a variety of different sensors to be connected to a mobile phone. The authors developed an architecture to dynamically interconnect sensors to a mobile application by generating a wrapper class. While the MoSHub approach is tailored for generating wrappers class for sensor devices, our approach is more general and intends to perform generic dynamic adaptation on mobile applications. Hence, dynamic configuration of sensor would be one use case for us. Authors [13] also do not discuss about features such as scalability, fault tolerance and distributed adaptation, for instance.

Another relevant work to us is the one proposed by [14]. The authors present an architectural model addressing flexible and adaptive composition of services in Very Large Scale (VLS) IoT systems by exploiting the concepts of service orchestration (i.e., centralized approach) and choreography (i.e., decentralized approach). While the authors follow a service orchestration/choreography model, which seems to be more adequate for web applications, we chose following a service-oriented component model [11]. Although the authors address VLS IoT systems, there is no information about how the architecture achieves scalability, and how the adaptation engineer defines the service composition.

MADAM (mobility- and adaptation-enabling middleware) [15] is a middleware to facilitate the development of adaptive mobile applications. MADAM employs an architecture-centric approach to allow parametric and compositional adaptations. It allows the adaptation engineer to implement utility functions that helps the MADAM to reason about the most appropriate application variant (i.e., application adaptation). One of the main differences between MADAM and our work is that the former runs only locally on the MNs without exchanging information among them (i.e., it is not distributed), whereas the latter employs a distributed architecture where the Adaption Manager, which is deployed within the SDDL core network, manages the adaptions performed by the MNs. One advantage of such approach is that we are able reason about the entire system, not only about a MN. Other differences are that our approach can

dynamically deploy new functionalities on the MNs, and manage distributed adaptation among MNs.

VI. Conclusion and Discussion

In this paper, we propose an approach to manage the complexity of building adaptive mobile application. Instead of managing low-level adaptation techniques (i.e., how to dynamically deploy new components or change parameters), we are focused in providing management of distributed dynamic adaptation and facilitating the development of adaptation policies. Hence, the main contribution of this paper is the proposal of (non-) transactional and (a) synchronous distributed adaptation for IoMT scenarios. We also present some preliminary performance results. Several studies have been conducted in the field of middleware for adaptive applications; however, most current efforts does not take into account problems such as mobility, scalability and manageability. There are many research challenges in this field; however, problems such as parametric variability, adaptation reasoner and adaptation mechanisms are not covered by our research.

We are aware that much work and research is needed, but we are confident that our approach is suitable to build CPS systems and will facilitate the development of such systems. However, we expect the following contributions in this and the next years: (i) an API tailored to develop mobile adaptive applications; (ii) the design of an interface to decompose the system in small and independent components; (iii) a mechanism to enable adaptation engineers to receive and handle events generated by the MNs; (iv) the design of (non-) transactional distributed adaptation; and (v) the design of asynchronous adaptation.

References

[1] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber-physical systems," *Concurr. Comput. Pract. Exp.*, vol. 23, no. 9, pp. 905–923, Jun. 2011.

[2] L. Gurgen, O. Gunalp, Y. Benazzouz, and M. Gallissot, "Self-aware cyber-physical systems and applications in smart buildings and cities," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 1149–1154.

[3] A. J. Ramirez and B. H. C. Cheng, "Design patterns for developing dynamically adaptive systems," in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10*, 2010, pp. 49–58.

[4] R. O. Vasconcelos, L. D. N. e Silva, and M. Endler, "Towards Efficient Group Management and Communication for Large-Scale Mobile Applications," in *5th International Workshop on Pervasive Collaboration and Social Networking (PerCol), co-located with Percom*, 2014.

[5] OMG, "OMG Data Distribution Portal," 2012. [Online]. Available: http://portals.omg.org/dds/. [Accessed: 08-Nov-2012].

[6] L. David, R. Vasconcelos, L. Alves, R. Andre, G. Baptista, and M. Endler, "A Communication Middleware for Scalable Real-Time Mobile Collaboration," in *IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2012, pp. 54–59.

[7] G. L. B. Baptista, M. Roriz, R. Vasconcelos, B. Olivieri, I. Vasconcelos, and M. Endler, "On-line Detection of Collective Mobility Patterns through Distributed Complex Event Processing," Monografias em Ciência da Computação - MCC 12/2013, Dep. de Informática, PUC-Rio, ISSN 0103-9741, 2013.

[8] R. O. Vasconcelos, M. Endler, B. Gomes, and F. Silva, "Autonomous Load Balancing of Data Stream Processing and Mobile Communications in Scalable Data Distribution Systems," *Int. J. Adv. Intell. Syst.*, vol. 6, no. 3&4, pp. 300–317, 2013.

[9] K. Kakousis, N. Paspallis, and G. A. Papadopoulos, "A survey of software adaptation in mobile and ubiquitous computing," *Enterp. Inf. Syst.*, vol. 4, no. 4, pp. 355–389, Nov. 2010.

[10] G. Jacques-Silva, B. Gedik, R. Wagle, K.-L. Wu, and V. Kumar, "Building user-defined runtime adaptation routines for stream processing applications," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1826–1837, 2012.

[11] C. Escoffier, P. Bourret, and P. Lalanda, "Describing Dynamism in Service Dependencies: Industrial Experience and Feedbacks," in *Proceedings of the 2013 IEEE International Conference on Services Computing*, 2013, pp. 328–335.

[12] R. O. Vasconcelos, M. Endler, B. de T. P. Gomes, and F. J. da S. e Silva, "Design and Evaluation of an Autonomous Load Balancing System for Mobile Data Stream Processing Based on a Data Centric Publish Subscribe Approach," *Int. J. Adapt. Resilient Auton. Syst.*, vol. 5, no. 2, 2014.

[13] C. Perera, P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Dynamic configuration of sensors using mobile sensor hub in internet of things paradigm," in *IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 473–478.

[14] K. Dar, A. Taherkordi, R. Rouvoy, and F. Eliassen, "Adaptable service composition for very-large-scale internet of things systems," in *Proceedings of the 8th Middleware Doctoral Symposium on - MDS '11*, 2011, pp. 1–6.

[15] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, "Using architecture models for runtime adaptability," *IEEE Softw.*, vol. 23, no. 2, pp. 62–70, Mar. 2006.