The Journal of Systems and Software 106 (2015) 9–27

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Engineering Future Internet applications: The PRIME approach

Mauro Caporuscio [a,*], Carlo Ghezzi [b]

[a] Department of Computer Science – Linnaeus University, Växjö SE-351 95, Sweden
[b] Dipartimento di Elettronica, Informazione e Bioingegneria – Politecnico di Milano, P.zza L. Da Vinci 32, Milano 20133, Italy

ARTICLE INFO

Article history:
Received 23 June 2014
Revised 30 March 2015
Accepted 31 March 2015
Available online 14 April 2015

Keywords:
Resource-oriented architecture
Middleware
Future Internet

ABSTRACT

The Future Internet is envisioned as a worldwide environment connecting a large open-ended collection of heterogeneous and autonomous resources, namely *Things, Services* and *Contents*, which interact with each other anywhere and anytime. Applications will possibly emerge dynamically as opportunistic aggregation of resources available at a given time, and will be able to self-adapt according to the environment dynamics. In this context, engineers should be provided with proper modeling and programming abstractions to develop applications able to benefit from Future Internet, by being at the same time fluid, as well as dependable. Indeed, such abstractions should (i) facilitate the development of autonomous and independent interacting resources (*loose coupling*), (ii) deal with the run-time variability of the application in terms of involved resources (*flexibility*), (iii) provide mechanisms for run-time resources discovery and access (*dynamism*), and (iv) enable the running application to accommodate unforeseen resources (*serendipity*).

To this end, PRIME (P-Rest at design/run tIME) defines the P-REST architectural style, and a set of P-REST oriented modeling and programming abstractions to provide engineers with both design-time and run-time support for specifying, implementing and operating P-RESTful applications.

1. Introduction

Since the first packet-switching network was deployed, the evolution of the Internet has been constant and produced a phenomenon that radically changed communications. While initially simply used to exchange data between hosts, today the Internet is essential for the development and provision of software resources (e.g., data, applications, services) distributed all over the world. Furthermore, wireless network technology (e.g., 4G, Wi-Fi Direct, and Bluetooth LE) allows autonomous mobile devices (e.g., sensors, smartphones, and real world objects) to both consume and provide software resources over the Internet. On the other hand, from the application-level perspective, the Internet evolution is characterized by a fast transition from "sharing" (Web1.0) and "contributing" (Web2.0) toward "contextualizing" (Web3.0), which encompasses machine-facilitated understanding and correlation of software resources.

Several EU[1] and USA[2] research initiatives are currently focusing on the definition and development of the Future Internet (FI). According to the vision of the EU research initiative (Papadimitriou, 2009), FI is built upon three key pillars – *Internet of Things, Internet of Services, Internet of Contents* – underpinned by a dynamic network infrastructure. FI is envisioned as a worldwide pervasive execution environment dynamically formed by interconnected real-world objects, which are all around us, everywhere and anytime, and can be discovered, correlated and consumed as needed. Indeed, FI can be considered as a worldwide network-based system,[3] where a large open-ended collection of heterogeneous and pervasive resources dynamically interact with each other, to provide users with rich functionalities, e.g., *content* sharing, *service* provisioning, and real *thing* consumption.

FI applications will dynamically emerge as opportunistic aggregation (Preda et al., 2012) of resources of interest available at any given time. To achieve this vision, key objectives are: (i) *Abstraction* to facilitate the design, implementation, and integration of software resources, irrespectively of their specific nature (i.e., *thing, service*, and *content*) and location, and (ii) *Contextualization* to support opportunistic discovery and correlation of resources of interest.

Concerning *abstraction*, software engineering best practices suggest the exploitation of middleware solutions, which mask the distribution and heterogeneity of both the execution and the networking environments, and support the development of network-based

* Corresponding author. Tel.: +46 470708558.
E-mail addresses: mauro.caporuscio@lnu.se (M. Caporuscio), carlo.ghezzi@polimi.it (C. Ghezzi).

[1] http://www.future-internet.eu.
[2] http://www.nets-find.net.

[3] Network-based systems rely on explicit distribution of independent and autonomous components, which interact by means of (asynchronous) message passing (Tanenbaum & Van Renesse, 1985).

http://dx.doi.org/10.1016/j.jss.2015.03.102
0164-1212/

systems through the provision of proper mechanisms for designing and implementing software resources, as well as for deploying, publishing, discovering and binding them at run time (Issarny et al., 2007). Existing middleware solutions provide different types of abstraction – e.g., procedure, object, component, service – to deal with these issues. However, all the stability assumptions made on network-based systems are no longer valid in the continuously changing contexts that characterize FI, where mobility affects the dynamic availability of resources (Roman et al., 2000). As a result, FI applications can be viewed as a dynamic software architecture where both the entities and their interconnections can change at run time. In the FI setting, applications are required to be "fluid". By this we mean that they must be able to accommodate continuous architectural changes, possibly without affecting their behavior (Kortuem, 2006). Therefore, key requirement is to provide developers with a set of proper abstractions enabling architectural fluidity: (i) *loose coupling*: entities are deployed and executed independently of other entities, (ii) *flexibility*: entities can be added and removed into the running application, (iii) *dynamism*: entities of interests are discovered and bound into the running application, and (iv) *serendipity*: unforeseen entities are accommodated into the running application.

The literature defines *context* as "any information that can be used to characterize the situation of entities, such as individuals, places, or objects, that are deemed relevant to the interaction between a user and an application, including the user and the application themselves" (Dey et al., 2001). The term "contextualization" indicates that FI applications should be built by dynamically aggregating resources related to a particular situation, and be able to adapt to the evolving situation in which they operate, such as the physical environment and the computational entities populating it or the device on which the application runs. The challenges related to *Contextualization* concern the ability of *discovering, understanding, selecting*, and *correlating* resources of interests. Contextualizing resources of interest in an open-ended world asks for mechanisms to semantically describe both functional and extra-functional properties of the resources, and to reason about them and their actual context. To this end, semantic description of software artifacts demonstrated to be effective and has been being largely adopted in the context of Semantic Web (Berners-Lee et al., 2001), which employs ontologies as main building blocks to enable the semantic web vision.

Contribution of the work – In this work we study FI from a software engineering perspective. Specifically, we develop a principled, middleware-supported architectural approach that is intended to support fluid applications. Our main contribution is twofold. First, we present a fully revised and extended version of the P-REST architectural style (Caporuscio et al., 2011), and define a set of *modeling abstractions* to provide engineers with design-time support for specifying P-RESTful applications.[4] Second, we define a set of *programming abstractions* to provide engineers with run-time support for implementing and operating P-RESTful applications. Indeed, the PRIME (P-Rest at design/run-tIME) approach provides both modeling and programming abstractions for (i) uniformly representing things, services and contents as *resource*, and (ii) developing FI applications as opportunistic aggregations of *contextualized resources*.

To assess the applicability of our approach, we evaluate PRIME's modeling and programming abstractions in supporting the development of a set of FI applications, namely *Smart City Pulse, Smart Slide Show*, and *Smart eHealth*. We also empirically evaluate the effectiveness of PRIME's middleware in supporting resource management, discovery, provision, and interaction.

The paper is structured as follows: Section 2 introduces a motivating scenario further used as running example throughout the paper, and Section 3 discusses related work. Sections 5 and 6 discuss design rationale for resource *abstraction* and *contextualization*, respectively, whereas Section 7 details their implementation. The assessment of PRIME is carried out in Section 8, which evaluates both efficacy and efficiency of the approach. Finally, Section 9 concludes the paper and sketches our perspectives for future work.

[4] That is, applications that conform to the P-REST style.

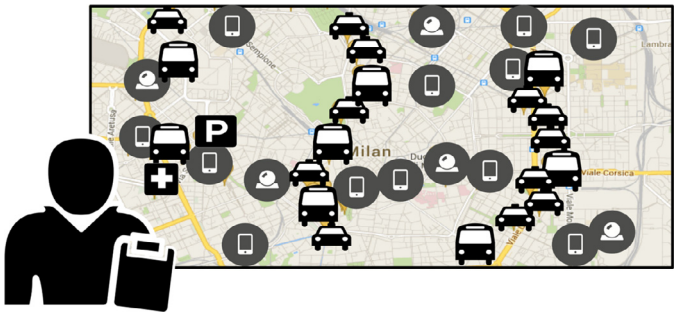

Fig. 1. Smart City Pulse application – scenario.

2. Motivating scenario

This section introduces *Smart City Pulse*, a Future Internet application that serves as running example throughout the paper to illustrate the proposed approach.

Smart City Pulse is a crowdsourcing application that aims at finding, retrieving, and monitoring data produced by a networked sensing infrastructure made of sensors embedded in end-user mobile devices (e.g., GPS, temperature), public utilities (e.g., smart metering, surveillance, traffic and pollution monitoring), and many other kinds of sensing systems. Referring to Fig. 1, *Smart City Pulse* is composed of: (i) an open-ended set of heterogeneous *Sensors* deployed within the city, and (ii) a *Monitoring Service* that analyzes the data gathered from the sensors.

Functional requirements for *Smart City Pulse* are specified as follows:

R0: *Sensors* are deployed on mobile devices, and may enter/leave the network dynamically.

R1: *Sensors* are autonomous and *the Monitoring Service* has no a-priori knowledge of them.

R2: The number of *Sensors* engaged in the application changes over the time.

R3: The types of *Sensors* engaged in the application change over the time.

R4: A *Monitoring Service* dynamically discovers and accesses new sensors. For instance: (**R4.0**) all *Sensors* in the network, irrespectively of their specific type; (**R4.1**) all *Sensors* of a given type (e.g., temperature); (**R4.2**) all *Sensors* within a given area (e.g., downtown); and (**R4.3**) all *Sensors* related to a given situation (e.g., weather report).

3. Related work

Related work spans several research areas, from pervasive computing to software architecture and semantic web. This section focuses on related work in the two areas where PRIME mainly contributes, namely *engineering resource abstraction* and *engineering resource contextualization*.

3.1. Engineering resource abstraction

Software engineering best practices suggest to support the development of network-based applications through the exploitation of middleware, which provides engineers with abstractions for designing and implementing networked software resources (Issarny et al., 2007).