COMPOSE — A journey from the Internet of Things to the Internet of Services

Benny Mandler
IBM – Haifa Research Lab
Haifa, Israel

Fabio Antonelli
CREATE-NET
Trento, Italy

Robert Kleinfeld
Fraunhofer FOKUS
Berlin, Germany

Carlos Pedrinaci
The Open University
Milton Keynes, UK

David Carrera
Barcelona Supercomputing Center
Barcelona, Spain

Alessio Gugliotta
Innova
Rome, Italy

Daniel Schreckling
University of Passau
Passau, Germany

Iacopo Carreras
U-Hopper
Trento, Italy

Dave Raggett
W3C
Bradford on Avon, United Kingdom

Marc Pous
BDigital
Barcelona, Spain

Carmen Vicente Villares
Abertis Telecom
Barcelona, Spain

Vlad Trifa
EVRYTHNG
Zurich, Switzerland

***Abstract*—The COMPOSE project aims to unleash the full potential harbored by the Internet of Things by creating a complete ecosystem around it to enable the flourishing of a resulting Internet of Services, seamlessly integrating the real and virtual worlds. COMPOSE will achieve this through the provisioning of an open and scalable marketplace infrastructure, in which smart objects are associated to services that can be combined, managed, and integrated in a standardized way to easily build innovative applications. The resulting platform is expected to significantly strengthen the service industry in Europe.**

Keywords—Internet of Things; Internet of Services; semantics; heterogeniety; scalability; security; composition

I. INTRODUCTION

The Internet of Things (IoT) has a huge potential to fundamentally alter the way in which we interact with the physical and virtual world around us, and pave the way for a brand new kind of services that will escort us wherever we turn. Nevertheless, this potential has not yet been realized due to the current complexity faced by potential exploiters. The level of complexity stems from the absence of several important building blocks. The current state of the art forces potential service providers to deal with the entire hardware and software stack required to realize such services. The effort needed starts from communicating with a heterogeneous set of things, through the creation of specific services and possibly the composition of several such services, through service deployment, adequate run-time and communication support, including non-functional requirements such as reliability, while maintaining adequate levels of security throughout the different layers.

COMPOSE aims to create an ecosystem for unleashing the power of the Internet of Things via an easy transformation to an Internet of Services. The ecosystem is targeted for different kinds of stakeholders, namely, "things", service creators, infrastructure providers, and end users. We aim to lower the barrier of entry to this market, thus opening the door for different kinds of players to participate, each according to its own expertise. For example, SMEs and individual entrepreneurs will be able to introduce innovative services to the mass market, while large enterprises will provide the basic infrastructure, and end-users will benefit from services that were previously economically unfeasible. The COMPOSE ecosystem is envisioned as an open marketplace of services for the Internet of Things.

The remainder of this paper is organized as follows. Section II provides an outline of the foreseen architecture. Section III introduces the objects virtualization layer. Section IV details the services management and deployment. Section V details the platform back-end infrastructure. Section VI highlights platform security aspects. Section VII introduces the open marketplace. Section VIII details the envisioned pilots and business models. Finally, section IX details plans for fostering a developer's community.

II. COMPOSE ARCHITECTURE

The logical architecture of the COMPOSE marketplace is depicted in Figure 1. An object is any physical active device capable of either providing contextual data or acting on the external environment, or a virtual representation thereof. This includes sophisticated objects such as mobile devices and multi-sensing platforms, but also simple ones like RFID tags.

COMPOSE implements a Service-Oriented Architecture, where any resource is provided and consumed in the form of a service. An Object is then elicited to a service object when it becomes accessible through a network connection, enabling programmatic interaction for obtaining data or carrying out a certain action. While an object would be the sensing device monitoring the status of a house, for example, its corresponding service object is the ICT abstraction of a given feature provided, such as data on the temperature inside the house. Service objects will comply to the COMPOSE standardized interfaces, and will be potentially running in the COMPOSE runtime environment in order to be accessed or actuated on.

Service objects can be standalone or composite. For example, a house service object can be a composition of various objects providing information on temperature, presence, light and more.

This research has been partially supported by the EU under the ICT Collaborative Project COMPOSE (Collaborative Open Market to Place Objects at your SErvice), grant agreement n. FP7- 317862

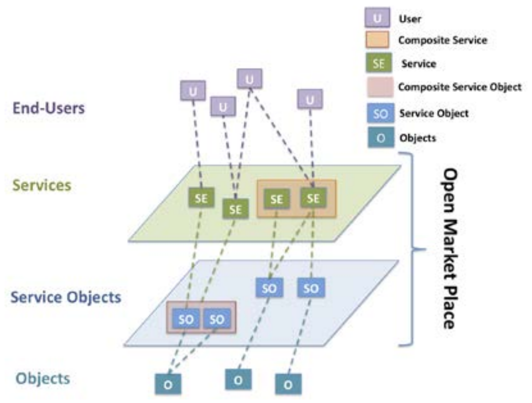

Figure 1: COMPOSE logical architecture

Composite service objects can provide information obtained from the aggregation of multiple data flows coming from different stand-alone service objects.

Services are ICT abstractions of software systems that provide a certain functional value that can be exploited to achieve a certain goal and are reusable and repurposable. The simplest kind of Services we contemplate in COMPOSE will be based on the direct exploitation of data from or about Service Objects (e.g., if the parking sensor is off, the space is free) thus providing a functional interpretation of Service Objects state and/or data. The same Service Object could be susceptible to diverse interpretations in different settings therefore being the basis for various services. A service can be both a consumer of information originating from service objects and an actuator connected to one or multiple service object(s). When acting as a consumer, a service uses the information originating from one or more service objects to perform a given task. In contrast, when actuating on a service object, a service issues a task to such a service object. For example, a service could first consume the information from a light sensor, and then determine whether to actuate the light.

End users are the consumers of services managed through COMPOSE. A user can be a person, accessing the marketplace through a personal device or computer, or a machine, through an appropriate machine-to-machine (M2M) protocol interface, interacting with the market to integrate IoT services into its business process. COMPOSE provides the distributed infrastructure orchestrating all aspects of the components mentioned above. Data coming from objects can be streamed into the marketplace, where their counterpart service objects will operate, and can be published such that other entities will be able to find the information and consume it. The marketplace will ensure that privacy and security aspects are well taken care of and additional non-functional requirements such as QoS may be specified.

Accordingly at the lower architectural level resides an objects virtualization layer, which harmonizes object and data access and ingest. Above it lays a services management layer which adds semantic to objects, and allows for service discovery, composition, and deployment. All aspects are hosted within an elastic run-time supported by scalable communication facilities, providing platform wide security.

III. Objects as a Service

Objects will be exposed as high-level services. For that purpose, each object will be represented in two repositories planned for COMPOSE: the object registry and the object data store. The former will store the object representation, basic information and semantic data. The latter will be used to store any object-generated data. Both data stores will have to offer flexibility for scaling as needed while delivering linear performance scalability. The data stored in both data stores will be retrieved through data digestion layers, that is, high-level services will get access to data management primitives that will provide customizable views of the data. For such purpose, basic Domain Specific Languages (DSLs) will be developed to express, with minimal complexity, such data processing primitives. At the same time, any objects generating real-time streams of data will be associated with real-time distributed data processing mechanisms that will allow for the storage of digested data only. In other words, the deployed services will be allowed to express data processing primitives that will transform object-generated raw data into the defined data model. The same DSLs mentioned above will be leveraged for that purpose. Interaction with physical objects will take place though the Web of Things approach: APIs based on web technologies (RESTful when possible) will be leveraged to sense and actuate the objects.

The object data stores, being a highly demanding component, pose an architectural challenge for COMPOSE. Thus a concentrated effort will be made to make them extremely scalable, extensible, and achieving high performance. To achieve such a goal, both data repositories will comprise a multi-level key/value storage design. In COMPOSE we will define multiple layers of key/value stores according to the characteristics of existing storage devices (i.e. RAM, Flash, and Mechanical disks). In-memory key/value stores will be combined with state of the art NoSQL technologies to produce such a novel architecture. New caching strategies will be explored to operate in fully distributed modes, providing critical reliability and scalability properties needed for the COMPOSE marketplace

Data management aspects, such as aggregation, filtering, and joining of data in the object store will be handled with cloud-scale data analytics solutions, such as MapReduce. Data management primitives will be expressed using newly developed DSLs, and will be deployed with the services. The COMPOSE framework will transform the primitives deployed with the services into data access strategies on top of the multi-level repositories. New advances will be required to extend such runtimes to support processing large amounts of data while observing QoS requirements imposed by objects and services. While existing work on data management and structuring exists, the goal of proposing new mechanisms for Internet-scale data management for a huge network of service objects poses new challenges for a multi-level architecture.

Services will also have the option to be deployed with associated real-time data processing primitives. Such primitives will be expressed using DSLs and will be translated into stream processing units that will digest object-generated raw data into data suitable to be stored. Changes in the services or deployment of services with different needs will have to be handled automatically. The data models selected to store object data will be enhanced to work with semantic data, keeping in mind that semantic composition will have to be supported. As a result, on-the-fly data model changes should be supported.

IV. SERVICE MANAGEMENT

A fundamental tenet of Service-Oriented Architectures is the notion of service registry for programmatic access and discovery of appropriate reusable services. Much effort has thus been devoted to developing advanced algorithms and infrastructure able to support developers and applications in efficiently and effectively discovering services online[1]. Despite the progress thus far, most solutions have typically been anchored on the use of Web Service Description Language (WSDL) for describing services and on the use of a centralized registry manually populated by developers [2]. COMPOSE will strive to provide a novel distributed and scalable infrastructure for the dynamic, efficient and effective discovery of highly heterogeneous services. At the lowest infrastructure level, the discovery engine will exploit the notion of a Distributed Hash tables (DHT) in order to provide a distributed and highly efficient look up system supporting continual node arrivals and departures as well as failures.

At a higher level the discovery engine will exploit Linked Services technologies and infrastructure [3] in order to be able to exploit lightweight semantic technologies for carrying out expressive yet scalable service discovery over highly heterogeneous services including Web services, Web APIs as well as data provisioning objects such as sensors and mobile devices. Semantic matchmaking algorithms will thus be extended and improved to provide efficient and scalable distributed discovery. Additionally, the infrastructure will be complemented with novel solutions for the automated identification of services and a best-effort approach to the automated generation of annotations and additional metadata characterizing the services at hand in order to provide semantic descriptions of the services found in terms of the kind of service, the data provided, and some general assessments about their reliability, the quality of their data, etc. To this end we shall devise novel information extraction and semantic annotation algorithms able to process whenever possible the existing descriptions of services including for instance their HTML documentation, as well as machine learning algorithms for the automated characterization of services based on the data provided and previously encountered examples.

The discovery of services by a user or an application has traditionally been an activity essentially triggered by the consumer, may this be at design time by a developer, or at runtime by an application using what is often referred to as late binding capabilities [2]. While this functionality indeed covers the most typical usage scenario, i.e., one whereby the developer or application knows exactly what type of service is required, it does not enable other highly valuable complementary usage scenarios. For instance, it does not promote or contribute in any way to application innovation since users are only driven by their initially anticipated objectives and therefore hardly explore the use of other services that could potentially be valuable for the objective sought [4]. Similarly, traditional discovery techniques do not directly cater for the continuous improvement of applications that could be achieved by continuously and dynamically changing the service(s) used on the basis of their currently exhibited behavior or simply the availability of more suitable services at a given moment. In the scenarios contemplated in COMPOSE situations where a service disappears at runtime, its service level degrades, or a new more adequate service is made available will be very frequent and must therefore be adequately accommodated. We shall devise a novel service recommendation engine that will provide continuous proactive advertising of services to users and applications based on the knowledge about i) the actual application being executed or developed, ii) the services used in the past in similar applications and others available in general, iii) previous actions of users. At the infrastructure level, the service recommender will rely on the highly efficient, distributed and scalable infrastructure with notification support in order to quickly determine situations where potential improvements may be obtained by replacing services, and it will subsequently trigger the appropriate notification to the application layer. At the component level the recommendation engine will combine content-based and collaborative filtering techniques over services, applications, and users in order to determine potentially interesting services both at design time and at runtime [5]. Notably, we shall leverage the information automatically derived about services while discovering them as well as previous usage of services in other applications in order to better characterize the services as well as the users and their preferences.

COMPOSE's main goal is to empower users in creating new applications out of existing services provided by Web APIs, sensors and mobile devices. The creation of these applications will most often involve combining different services in order to provide some added value functionality ensuring that the resulting compositions are secure and can quickly be adapted to react to changing environments and conditions. In the area of service-oriented computing the manual composition and ulterior orchestration of services has perhaps been the primary objective and has led to a number of composition languages, e.g., BPEL, BPMN, and related execution engines [1]. There have been also considerable efforts in assisting or automatically supporting the generation of compositions given the available situation and the desired objective [5]. Despite the outstanding progress thus far, the current state of the art falls short when it comes to dealing with such a highly demanding scenario where a large and highly variable number of services encapsulated behind highly heterogeneous interfaces will need to be efficiently (re)combined. On the one hand many solutions exploit solely syntactic services descriptions based on WSDL which in our case will hardly be available and are in any case not enough to ensure the compatibility of the resulting compositions. On the other hand, richer proposals able to exploit the semantics of services rely on expressive formalisms and rich service annotations which make them impractical in Web-scale settings both from a computational perspective as well as from a knowledge acquisition point of view. In order to cater for the dynamicity, scale and heterogeneity faced in the project we shall develop an assisted service composition engine able to provide efficiently semantically compatible service compositions of highly heterogeneous services. To this end, we shall i) leverage the use of Linked Data [5] for capturing service descriptions and sensor networks, ii) devise a novel approach to service composition based on the opportunistic interleaving of traditional semantic composition algorithms with our efficient dynamic semantic service discovery, iii) enhance the composition algorithm with advanced caching and dependency indexing mechanism to ensure an efficient execution, iv) combine the system with our advanced security analysis infrastructure in order to automatically provide an assessment of the level of security of the composition.

Once services are created, they need to be deployed. We aim to design and implement innovative models and mechanisms for the automated and distributed deployment of services. COMPOSE expects to provide new solutions to achieve management and monitoring of services in the marketplace. Services management will have deployment and operational aspects. COMPOSE will define and implement new methods for service publishing and provisioning in the marketplace and will also define and implement the methodologies and tools for automatic service construction. Specifically a study will be conducted on specifying and providing a virtual service execution. Moreover, we will define interfaces needed for proper services management touching upon services lifecycle, creation, upgrade, reconfiguration, resolving security conflicts, rerouting, etc. An accompanying monitoring component will oversee that SLA, security and privacy criteria's and QoS guarantees are met. COMPOSE expects to manage the lifecycle of services in the marketplace and will provide methods for on-the-fly provisioning of service components with "better" characteristics. The monitoring component will provide mechanisms for service provisioning, consumption monitoring and service lifecycle management

V. Scalable infrastrucre

The envisioned platform requires a highly scalable back-end infrastructure providing the run-time environment for objects, services, and application, as well as communication mechanisms to connect the varying entities.

A suitable run-time for the COMPOSE platform needs to be highly scalable and dynamic and handle a heterogeneous set of entities running within it, such as simple and composite services and service objects. The Runtime will comprise of a cloud environment and a will have a mobile extension, deployed over Smart Objects. Together they will be responsible for platform and services management, including discovery, QoS, monitoring, dynamic composition, as well as the dynamic control of services residing on Smart Objects. Moreover the runtime will need to be able to handle large amounts of data flowing into the platform from a plethora of objects and services. As part of its core functionality the runtime will be responsible for instantiating, deleting, and running platform entities, as well as enabling services and objects discovery. The runtime shall incorporate a notification service for subscriber entities running within or outside the scope of the platform. Such notification can be based on available smart objects, services, or data stemming from these entities. We plan to involve smart objects processing power and storage for hosting some COMPOSE entities, in addition to attempting to co-locate computation with data as close to the source as possible.

In parallel, the platform calls for the design and development of a scalable and fault-tolerant communication network to (i) connect all marketplace entities, (ii) support service discovery and lookup, (iii) share information. We will provide a highly scalable clustering infrastructure, based on peer-to-peer technologies. In order to reach the scale anticipated we aim for eventual consistency guarantees in conjunction with a hierarchical design. This component will support application fault tolerance in conjunction with a platform promoting high availability schemes including proper placement and load balancing.

A "grow as you go" design is envisioned such that the infrastructure will work equally well from small to large scale, and will not assume only a large scale design point. Our design calls for increased scalability without paying a performance penalty, including fast failure detection and notification, and handling gracefully multiple simultaneous failures. Several services will be exposed to provide a great deal of autonomy to the different components. Services envisioned include a membership service (including fast and efficient failure detection), and group communication services such as scalable publish / subscribe and a DHT. Services provided by this layer will be readily available for use by the runtime component, which will allow it to take more informed decisions concerning issues such as components replacement. Thus, all services combined will portray a unified source of control, management, and information distribution. We will focus on designing a system that automatically adapts to change, requires almost no manual operations, and simplifies objects addition/deletion to reduce the management overhead and ensure a sufficient level of scalability. Finally, we plan to handle heavy-churn scenarios, since service objects may join/leave the system at an unpredictably high rate.

VI. Security

Security requirements of different types of COMPOSE users strongly depend on the interaction with the platform. For example, for enterprises which provide specific input or computational resources to the system, COMPOSE must prevent unauthorized access to data or resources, provide accounting, identity management, etc. Individuals feeding private information into the system or consuming services have a growing interest in the secure processing of their data as well as a secure execution of potentially malicious services. Thus, the adoption of the COMPOSE marketplace strongly depends on the trust attributed to the platform. Therefore, feasible and comprehensive security systems are essential.

COMPOSE flexibility, dynamicity, and openness calls for a new set of security mechanisms. The envisioned platform will not disregard the need for node and communication security, appropriate access control mechanisms, as well as a consistent and holistic identity management. Projects such as Webinos [7] which deploy established security standards will serve as a basis for our work. They will be enhanced by a reputation framework which will address misbehaving nodes and services.

The power of COMPOSE strongly relies on the processing of data by services or service compositions. Services can generate data, service compositions can deliver data from different sources, and complex service compositions which can change during runtime, process this data and forward it to data sinks such as actuators. Furthermore, data and services are fed into the system by users with different security requirements, varying interest, and expertise. COMPOSE processes unknown data by unknown applications. This is critical for the enforcement of data as well as platform and application requirements. To address this challenge, COMPOSE will define new semantic modeling of data, APIs and service interfaces. Our models will specify usage and security constraints a user or company needs to define on objects, data, or services. This specification language forms the basis for a data provenance framework and emphasizes the data and information flow focus of the COMPOSE security framework.

Related projects[8][9] analyze services for security and in particular for security flow properties. However, they are based on static characteristics described by pre-defined security patterns. They do not draw dynamic data properties into consideration. Hence, the mechanisms used in these projects can only partially map to the requirements of COMPOSE. COMPOSE will deploy a combination of static analysis methods and runtime monitors. Static analysis will be used to automatically derive security annotations for services and API methods in SDKs. They will also help to develop efficient algorithms identifying insecure or critical data flows in services and service compositions. This information will then be used to support automated and secure service composition, service re-orchestration due to changes in the runtime environment of a service, and to support developers in implementing secure services. Based on the reports generated by the static analysis processes, we will also deploy model or code rewriting mechanisms which adjusts service implementations to (i) directly respect data security properties or (ii) integrate or trigger runtime monitors into the service implementation. These inline monitors account for the un-decidable security properties detected by the static analysis framework. They will also increase precision and performance of the usage control enforcement in the runtime environment.

The security framework in COMPOSE must be scalable. The analytical solutions and runtime security framework will be designed as a trade-off between performance and precision and will provide appropriate solutions to reduce the induced overhead. A certification framework for flow properties which is based on proof-carrying code will help moving the computational burden from the user devices to the marketplace. Further, it provides trust to the user executing services from the marketplace. Finally, we envision an adaptation of this paradigm to make the proof independent of user security requirements. To additionally reduce the analytical overhead induced by the magnitude of potential service compositions, we will introduce a variability aware analysis, efficiently digesting a large number of service compositions and their deployments on various user devices, complementing traditional security concerns with emphasis on data and information flows.

VII. THE COMPOSE OPEN MARKETPLACE

Marketplaces and app stores have become the new rage in this evolving era of technology and lifestyle. In this diverse environment COMPOSE will enhance the state of the art in app stores for native applications by providing an environment in which not only applications but also service objects and services can be traded. COMPOSE promotes a novel concept of trading services and service objects by sharing their interfaces between 3rd party applications which are able to run in the context of other users who are not the owners of the service object itself. COMPOSE will provide software components for managing the access rights for each smart object connected to the marketplace and will give the owner of the object full control to monitor the history of each application that accesses the service object and to update the associated rights and rules for accessing the service at any time.

The COMPOSE open marketplace is expected to contribute to the creation of a new generation of app stores that will enable developers to create services interacting with, and acting on, the real world through service objects. COMPOSE will address cross-device apps as mashup of service objects and services to be used not only on smartphones, but also on desktop computers, set-top-boxes or TV sets. The challenge, which applies equally to users, developers and Consumer Service Providers (CSPs) of an app, is to support as many terminal devices as possible. A user would like to access individual services on all of his devices whereas the CSP would like to make the service available with as little operational effort as possible. COMPOSE will facilitate the convenient creation, administration, distribution, and updating of apps for a variety of terminal devices. Thus, the COMPOSE open marketplace can be accessed via smartphones, set-top boxes / TV sets as well as tablets and desktop computers of different manufacturers to use apps, rate and purchase them, as well as to get recommendations.

COMPOSE will provide a cross-platform presentation layer for the open marketplace. The frontend of the open marketplace will provide Graphical User Interfaces (GUIs) for a variety of terminal devices. Furthermore, the open marketplace GUI will be designed specially for users without any programming experience. To access the open marketplace, users won’t need to install any software clients. The open marketplace will be accessible from anywhere using a traditional Web browser, using Web technologies (HTML5, JS, CSS).

Finding new apps involves time and effort in today’s app stores. This issue is of particular interest for developers searching for sensor data, linked data and services in repositories of IoT platforms. COMPOSE will ease this by offering smart search for apps but also for service objects and services. The marketplace will use app ratings, reviews, tags, and usage data to offer more than just a keyword-based search like today’s stores. Therefore, COMPOSE will exploit modern recommendation algorithms, social and semantic technology, and interactive data visualization schemes to provide interactive, fun, and innovative ways to discover relevant apps.

COMPOSE will provide a Software Development Kit (SDK) and an Integrated Development Environment (IDE) for developers that contain the necessary tools and Application

Programming Interfaces (APIs) to implement services and apps for the open marketplace. The IDE will provide tools that support developers in virtualizing real-world objects such as camera, RFID tags, traffic sensor, etc. In turn, these will be deployed as service objects in the COMPOSE open marketplace. From a business perspective, COMPOSE will enable app developers to define flexible feature-based access policies for their apps as a way to monetize their provided service. Thus, developers will be able to integrate flexible licensing with their apps. In addition to established licensing schemes, a CSP can define flexible licensing models for promotional and other purposes.

VIII. The COMPOSE Pilots

In order to offer reference business models for the marketplace and quantitatively assess its business potential, COMPOSE will analyze the marketplace business context and develop supporting application scenarios for piloting the marketplace in realistic settings. As the marketplace platform is aimed to be domain independent and to maximize outcomes of its business potential assessment, pilots identified for the platform and business model assessment will target a heterogeneous range of application scenarios, each of them having different scales (number of involved objects and different geographical coverage) and different target stakeholders. In particular: i) "Smart Spaces" use case will pilot COMPOSE in a shopping environment, focusing on the dynamic composition and delivery of services; ii) "Smart City" use cases will create a group of services for city citizens, focusing on the opportunities derived from the diversity of deployed sensors and from available open data; iii) "Smart Territory" will pilot the marketplace in a geographically broad territory in Italy (in the Trentino region), focusing on tourism and territory monitoring services.

The first pilot will focus on shopping environments as "smart spaces" where consumers, interacting with digitally augmented products, can benefit from an augmented shopping experience, enabling customers to get much more out of the products they buy, while manufacturers and retailers may directly link to their customers. This use case foresees the realization of a mobile application able to trace product information via an attached ID tag (e.g., NFC, RFID). Each product will be linked to the COMPOSE marketplace, and will be associated with a set of services, such as related offers. Such services may trigger a novel and personalized shopping experience, and improve brands marketing effectiveness.

The second pilot will focus on a "smart city" environment, using infrastructure deployed in Barcelona, the Abertis "SmartZone" [10], and OpenData project [11]. The focus will be on integrating heterogeneous devices and technologies, searching cross-data opportunities while using the wide set of existing sensors to develop and deliver new services for citizens to enhance day-by-day living in a city (e.g., integrated mobility, environmental and enhanced energy saving services).

The third pilot will focus on a "Smart Territory", deployed in a geographically broad scale in Trentino, and will use COMPOSE to create personalized socially/environmentally-aware territory monitoring and tourism services. These services will leverage the widely distributed broadband networking and environmental infrastructures available and the use of smart phones as service objects. COMPOSE will deliver personalized service mashups allowing tourists to explore the sports and cultural touristic facilities of Trentino, their actual climatic conditions, maps and timetables for transportation, queue information, and, leveraging the sensing capabilities of smart phones, offering sporty tourists a number of real-time services such as organization of public and private virtual competitions with other tourists, and receiving recommendations according to their level, current weather conditions and other criteria.

IX. Community Building and Training

A concentrated effort will be taken towards the creation of a community of interest around the COMPOSE platform. This community of developers will be pursued with the objective to stimulate the collaborative ideation and identification of possible exploitation paths and applications showing high business potential. We may start by approaching existing related communities (such as the "Web of Things", mobile development communities, etc.) and connecting with a meaningful number of key actors (experts, developers, professionals). Such a community will allow validating the platform and collecting from an early stage feedbacks on usability, effectiveness, and measurements on development time and cost reduction in the realization of applications integrating different types of data and objects. In turn that will contribute to the longevity of the developed platform.

Training sessions and materials capable of enhancing knowledge of COMPOSE services, development tools and their usage (APIs for accessing the marketplace, SDK for development), together with hands-on development sessions will allow the creation of real prototypes in meaningful test environments (e.g., the "Smart Zone").

References

[1] Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., "Service-Oriented Computing: State of the Art and Research Challenges", Computer, 40, 38–45, 2007.

[2] Pedrinaci, C., Domingue, J., "Toward the Next Wave of Services: Linked Services for the Web of Data", Journal of Universal Computer Science. 16, 1694–1719, 2010.

[3] Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecký, J., Domingue, J., "iServe: a linked services publishing platform", Ontology Repositories and Editors for the Semantic Web Workshop at The 7th Extended Semantic Web, 2010.

[4] Ngoc Chan, N., Gaaloul, W., "Collaborative Filtering Technique for Web Service Recommendation Based on User-Operation Combination", On the Move to Meaningful Internet Systems, 2010.

[5] Dustdar, S., Schreiner, W., "A survey on web services composition", Int. J. Web Grid Serv. 1, 1–30, 2005.

[6] Bizer, C., Heath, T., Berners-Lee, T., "Linked Data - The Story So Far", International Journal on Semantic Web and Information Systems, 2009.

[7] Fraunhofer FOKUS (ed.). Webinos Phase 1 Security Framework, 2011.

[8] AVANTSSAR: Automated Validation of Trust and Security of Service Oriented Architectures, FP7-ICT-2007-1, Project No. 216471.

[9] A. Yautsiukhin (ed.). D3.3 - Run-time Secure Composition and Adaptation Realisation Techniques, Dec. 2012.

[10] http://www.abertis.com/dyndata/301111_Smartzone_en_1.pdf

[11] http://w20.bcn.cat/opendata/