SICS Software-Intensive Cyber-Physical Systems
https://doi.org/10.1007/s00450-019-00413-w

SPECIAL ISSUE PAPER

RADON: rational decomposition and orchestration for serverless computing

G. Casale[1] · M. Artač[2] · W.-J. van den Heuvel[3] · A. van Hoorn[4] · P. Jakovits[5] · F. Leymann[4] · M. Long[6] · V. Papanikolaou[7] · D. Presenza[8] · A. Russo[1] · S. N. Srirama[5] · D. A. Tamburri[3] · M. Wurster[4] · L. Zhu[1]

Abstract
Emerging serverless computing technologies, such as function as a service (FaaS), enable developers to virtualize the internal logic of an application, simplifying the management of cloud-native services and allowing cost savings through billing and scaling at the level of individual functions. Serverless computing is therefore rapidly shifting the attention of software vendors to the challenge of developing cloud applications deployable on FaaS platforms. In this vision paper, we present the research agenda of the RADON project (http://radon-h2020.eu), which aims to develop a model-driven DevOps framework for creating and managing applications based on serverless computing. RADON applications will consist of fine-grained and independent microservices that can efficiently and optimally exploit FaaS and container technologies. Our methodology strives to tackle complexity in designing such applications, including the solution of optimal decomposition, the reuse of serverless functions as well as the abstraction and actuation of event processing chains, while avoiding cloud vendor lock-in through models.

Keywords Function as a service · Serverless computing · DevOps · Software models

1 Introduction

The success of IT-driven companies, like Apple, Facebook and Oracle, has shown that businesses investing in advanced software technologies can build a strong lead across a variety of market domains. A recent development with the potential to radically evolve the software technology landscape is serverless computing, a cloud-computing execution model proposed to entirely bypass user involvement in managing compute resources [1]. Function as a service (FaaS) is as of today the most prominent example of serverless technology. Using FaaS, individual function calls can be served remotely and automatically from the cloud [2]. This powerful idea brings us at a crossroads where current cloud software architectures are changing rapidly, increasing in flexibility and adaptability. In response, we propose a research agenda for developing an innovative DevOps framework centered around serverless computing, unlocking the advantages of the FaaS paradigm to industry.

To better understand these advantages and the implied research challenges, it is important to realize that the sudden popularity of platforms such as AWS Lambda,[1] Google Cloud Functions[2] and Microsoft Azure Functions,[3] may be attributed to the following two main promises of the serverless FaaS paradigm, among other benefits:

(1) *Fine-grained autoscaling* This refers to the ability to virtualize the internal logic of cloud-native applications, so that individual function calls are served remotely from the

This paper has been partially supported by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 825040 (RADON).

✉ G. Casale
g.casale@imperial.ac.uk

1 Imperial College London, London, UK
2 XLAB, Ljubljana, Slovenia
3 Jheronimus Academy of Data Science, 's Hertogenbosch, The Netherlands
4 University of Stuttgart, Stuttgart, Germany
5 University of Tartu, Tartu, Estonia
6 Praqma, Oslo, Norway
7 Athens Technology Center, Chalandri, Greece
8 Engineering Ingegneria Informatica, Rome, Italy

1 https://aws.amazon.com/lambda.
2 https://cloud.google.com/functions.
3 https://azure.microsoft.com/services/functions.

cloud and can thus harness autoscaling on demand. That is, serverless FaaS adds compute resources only to the portion of code that consumes them, simplifying scaling and saving costs compared to container as a service (CaaS) or other paradigms. Transparent and fine-grained scaling is particularly efficient for event-centric systems, as for example in the Internet of things (IoT), where actions need to promptly take effect in response to events triggered by code, data or the physical world.

(2) *Productivity gains* These arise from storing IT know-how as reusable serverless functions, which can be automatically orchestrated onto either private or public cloud infrastructures. Contrary to web services, serverless platforms automatically take care of resource management in a portable and reproducible way. Thus, every organization can be digitally transformed through investment in archiving core functions to be later used as building blocks for IT-driven business processes. Further to this idea, common functions may be shared in public marketplaces or repositories to establish a rich and diverse ecosystem. Such a development can considerably accelerate the delivery of new offerings to the market, even for companies with limited IT skills or no operations teams.

Despite the above promises, today there is still a lack of methodologies to reap the benefits of serverless FaaS in industry. Our vision is that open-source tools can greatly assist companies, especially small- and medium-sized enterprises, in building proof-of-concept and business cases with serverless FaaS at low costs, in order for them to responsively and progressively acquire technical capacity needed to incorporate this innovation in existing or new products. From an engineering point of view, a DevOps framework integrated with holistic management solutions that covers the entire life cycle of FaaS-based applications is still an open research problem that deserves attention from the software engineering community.

The research agenda of the RADON project (http://radon-h2020.eu, 2019–2021, funded by Horizon 2020) focuses on the development of an advanced DevOps framework for designing, prototyping, deploying, testing, verifying and evolving complex applications built on (i) serverless FaaS, defining events, triggers and actions (handling functions); (ii) services of various granularities, typically microservices, implementing business logic; (iii) data pipelines, i.e., specialized microservices (or serverless functions), resources and communication mechanisms in combination that manage the life cycle (ingestion, filtering, transformation, buffering, scheduling, analysis, transfer and storage) of data. The framework we aim for will support a continuous spectrum of service granularities, the fine-grained control of autoscaling at the function level, and deployment on infrastructures with heterogeneous capacities and at different locations, which are mediated by models.

2 Major challenges

In this section, we discuss the major challenges identified for the RADON research agenda, highlighting technical problems that are currently open and needs to be solved.

2.1 FaaS-oriented modeling

Each element in a microservices architecture constitutes a single, independent, functional and self-managing building block, e.g., a business logic module [3]. Serverless computing, particularly its instantiation into FaaS, is realized as a suitable programming paradigm for microservices, making the management of compute resources completely transparent and enabling the services to be designed in an event-centric manner [4]. Specific languages are emerging to natively program architectural design of applications composed of microservices, e.g., Jolie.[4] Such languages are a positive development but pose new problems. They typically rely on a custom interpreter that potentially carries the risk of vendor lock-in. Also, none of them support the specification of dependencies at the function level or the definition of data pipelines needed for business logic where events are essentially triggered by data. To counteract vendor lock-in, we argue that a model-driven approach can be combined with DevOps to deliver FaaS-based application. One major challenge in the RADON research agenda is then to introduce a FaaS-oriented modeling language with novel constructs for specifying function dependencies and defining data pipelines carrying events that trigger the function calls.

2.2 Requirements formalization

Common features of modeling languages include abstract syntax, i.e., the definition of concepts and how they relate to each other, and semantics, i.e., machine-interpretable mappings from abstract concepts to concrete elements in a given domain. The semantics of cloud modeling languages often reflects English prose [5]. This results in a shortage of formal solutions for synthesizing requirements across different layers of cloud applications, creating a gap with respect to advanced tools for automated synthesis of requirements arising from business goals and user scenarios outside the cloud context [6,7]. DevOps and continuous delivery (CD) also call for extension to existing formalization methods to simplify decomposition, synthesis and refinement of requirements up to the orchestration layer. In particular, there is a strong expectation for requirements formalisms that, while remaining close to concepts and abstractions typical of human thinking, are easy to translate into standard models for orchestration. To address these challenges, RADON will

[4] https://www.jolie-lang.org.

introduce a constraint definition language (CDL) for formally specifying both functional and non-functional requirements, increasing automation in designing FaaS-based applications.

2.3 Continuous integration and continuous delivery

Continuous integration and continuous delivery (CI/CD) are concepts that underlie agile approaches for software engineering [8,9]. They define full development and release pipelines that involve code compiling, functional testing, deployment packaging, artifact generation and storage, installation in test environments as well as delivery of complete artifacts. At the bottom level are configuration management tools, such as Chef,[5] Ansible[6] and Puppet,[7] that use an infrastructure-as-code (IaC) language to install and configure application components on the target infrastructure. Operating at the next level are orchestrators that take the deployment blueprint of a whole application, provision compute resources needed for running the application, and invoke configuration management tools to populate the resources. OpenTOSCA,[8] Cloudify[9] and Apache ARIA TOSCA[10] are examples of orchestrators that consume variants of OASIS TOSCA,[11] while Apache Brooklyn[12] and Juju[13] are based on other languages. At the top level are servers like Hudson[14] and Jenkins,[15] which are open-source, and commercial ones like TeamCity,[16] Bamboo,[17] Travis CI[18] and CircleCI.[19] These offerings have in common that users can set up CI/CD jobs to execute a given script upon certain events. For the RADON research agenda, the major challenge in this aspect is to extend existing solutions with capabilities to orchestrate serverless functions, microservices and data pipelines in a native, reusable and portable fashion.

[5] https://www.chef.io.

[6] https://www.ansible.com.

[7] https://puppet.com.

[8] https://www.opentosca.org.

[9] https://cloudify.co.

[10] https://ariatosca.incubator.apache.org.

[11] https://www.oasis-open.org/committees/tosca.

[12] https://brooklyn.apache.org.

[13] https://jujucharms.com.

[14] http://hudson-ci.org.

[15] https://jenkins.io.

[16] https://www.jetbrains.com/teamcity.

[17] https://www.atlassian.com/software/bamboo.

[18] https://travis-ci.org.

[19] https://circleci.com.

2.4 Quality assurance

Tools for functional testing are already an integral part of today's CD pipelines, e.g., automated unit and integration tests. However, testing non-functional properties, e.g., performance, has always been difficult [10,11], and additional challenges arise in the context of microservices [12]. An often-cited one is the trade-off between rapid delivery and extensive testing. One promising approach for this is to use monitoring data recorded in production to automatically create, select and evolve test cases to be executed through CD pipelines [13]. Among existing efforts, ITEA3 TESTOMAT[20] focuses on test automation in agile development but does not cover the microservices architectural style. ContinuITy[21] aims to improve the quality of load testing for microservices by evolving test specifications using monitoring data. DICE QT[22] automates the load testing of data pipelines by generating workloads that are stochastically similar to input traces. RADON is expected to offer a set of quality assurance tools for seeking latent defects in FaaS-based applications and verifying their satisfaction of service-level agreements (SLAs) and other non-functional requirements.

As in the DevOps quality assurance paradigm [14], IaC enables developers and operators to jointly create, configure and manage infrastructures by means of executable code, e.g., using a combination of the TOSCA language and configuration management tools like Chef, Ansible and Puppet. At present, the quality of IaC code is still largely dependent on the experience of developers and operators. A great deal of research work has been devoted in traditional software engineering to devising quality measures for the technical debt of code in recognition of for example bad smells [15,16], which are suboptimal design decisions made by developers that can affect the overall maintainability of a software system and make it more defect-prone [17]. However, none of technical debt management techniques, e.g., bad smell detection and defect prediction, have been applied to IaC languages. This is a gap that needs also to be filled by the quality assurance tools to support DevOps, broadening the spectrum of challenges in the RADON research agenda.

3 RADON framework

To tackle the identified challenges, we propose a research agenda for an integrated DevOps framework. The authors of this paper, who are involved in the RADON project, will

[20] https://www.testomatproject.eu.

[21] https://continuity-project.github.io.

[22] https://github.com/dice-project/DICE-Quality-Testing.

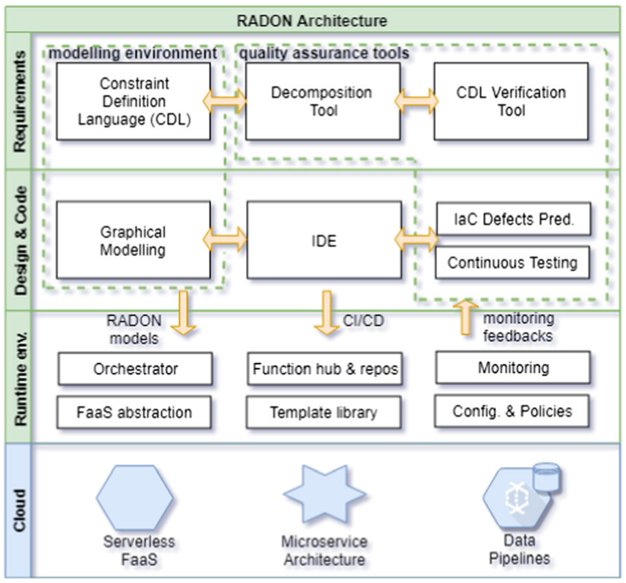

Fig. 1 Architecture of the RADON framework

continue to concretely deliver on this proposal and release the results as open-source software.

The RADON framework is envisioned to comprise three environments: (i) the modeling environment, (ii) the runtime environment, (iii) the coding environment (IDE). It will be coupled with a DevOps methodology to coordinate the development and release of FaaS-based applications, and with quality assurance tools to validate a particular solution. Figure 1 illustrates the architecture of the framework.

3.1 Modeling environment

RADON will rely on a model-driven approach for creating and managing cloud applications that typically exploit the microservices architectural style and the serverless FaaS paradigm. We see OASIS TOSCA as being in the best position to act as the baseline modeling language to describe the topology and orchestration of such applications, on top of which a novel family of RADON models will be defined. At present, no native support is provided by the TOSCA language for serverless FaaS or data flows, which however are both critical to RADON's significance due to the fact that serverless functions are often used to handle events triggered by data, e.g., real-time streams and diagnostic logs. Preliminary work has been done on the extension to TOSCA for the modeling and automated deployment of FaaS-based applications [18]. This is considered as a basis for our further research in RADON.

One conceptual difference between RADON and TOSCA models lies in incorporating the behavior specification of FaaS-based applications. For example, RADON tends to simplify the description of a temporal sequence of actions triggered by a certain chain of events via graphical annotations, while in cases where the graphical complexity may be a limiting factor, users will have the possibility to annotate temporal predicates through a CDL. This is an intuitive logic-based language to be introduced in RADON for specifying temporal behavior and formal requirements (for performance, costs, security and privacy) at different development stages of an application. The CDL will enable users to relate entities as well as events in RADON models, including those pertaining to data that transit on a pipeline. With CDL annotations, RADON models can retain two core benefits of TOSCA: (i) graphical and textual modeling, thanks to the Eclipse Winery project[23] and the TOSCA YAML specification;[24] (ii) automated orchestration of cloud applications at runtime.

3.2 Runtime environment

The RADON runtime will package tools that bridge the modeling environment and IDE of the framework with the cloud, relying on model-driven orchestration and IaC to enact the deployment of FaaS-based applications and data pipelines on multiple target cloud platforms. Particularly, development in the absence of an operations team will also be taken into account in this methodology, so that the development team can manage the runtime life cycle of an application in a self-service fashion. Model-to-text transformation between graphically built models and TOSCA YAML files will be performed by Eclipse Winery to directly feed the RADON orchestrator.

The runtime environment will feature a template library that encapsulates at the proper levels of abstraction all the necessary elements of microservices architectures. This will extend the DICE TOSCA library,[25] a baseline that encodes many TOSCA templates for Big Data processing. The extension for FaaS-based applications includes the node representations of individual functions and relationship representations that express dependencies as well as annotations for events foreseen to trigger a function. In effect, a template defines a directed acyclic graph (DAG) of components that make up an application, inclusive of those pertaining to serverless computing. Purpose-built microservices, called event gateways, will serve as mediators between event producers and consumers across multiple serverless FaaS platforms, realizing an abstraction layer internal to the application. Developers will be able to implement functions at their discretion and store the implementations in a function hub provisioned within both the runtime environment and the

[23] https://projects.eclipse.org/projects/soa.winery.

[24] https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML.

[25] http://dice-project.github.io/DICE-Deployment-Cloudify.

graphical modeling tool, allowing the reuse of business logic according to a function life-cycle model devised in RADON.

A special focus will be put on enabling data engineers to carry out the control (ingestion, buffering, scheduling, transfer and storage) as well as the processing (filtering, transformation and analysis) of data across FaaS-based applications using data pipelines, which are essentially composed of specialized microservices (or serverless functions), resources and communication mechanisms. Data pipeline templates will be developed as part of the template library, combining these building blocks into reusable TOSCA components that can be injected into RADON models and consumed by the orchestrator.

Upon deployment of an application, the RADON orchestrator will leverage IaC at the configuration management level to set up and wire up all the components. It will automatically configure monitoring services and connect metric collectors with user-specified storage, which can primarily make passive monitoring useful for the quality assurance of the application but also enable the orchestrator to adhere to performance requirements defined in the template. The orchestrator will be able to dynamically reconfigure the application components and change the number of node instances according to the scaling policies, while the serverless FaaS platform handles the autoscaling of the functions.

The security and privacy policies, e.g., access control and data encryption, will be implemented by the RADON orchestrator to manage the application components at runtime and their initialization process as well in order to limit the exposure of critical functionalities, ensuring that both business logic and sensitive data are protected even before the application starts running. The orchestrator will exploit service meshes to enact the security and privacy policies for FaaS-based applications, which consist of serverless functions, microservices and data pipelines. In essence, a service mesh forms a separate control layer for managing and configuring intelligent proxies deployed as sidecars that mediate and route traffic among services.

3.3 IDE & DevOps methodology

RADON models will be exposed via a web-based graphical IDE (Eclipse Che[26]). By clicking corresponding graphical elements, users will be able to (i) define serverless functions and microservices (for developers); (ii) design IaC recipes such as TOSCA YAML files (for runtime operators). Data engineers will also rely on the IDE to implement data transformations as serverless functions or microservices and combine these building blocks into reusable data pipeline templates. Under teamwork circumstances, users with different roles will be granted an appropriate level of access.

[26] https://www.eclipse.org/che.

A DevOps methodology will be investigated, identifying stakeholders, the socio-technical system, barriers to adoption, customization methods, recurrent architectural patterns and the roles of users interacting with the RADON framework. An important feature of the methodology will be the conceptualization of different life cycles at play for the components of a FaaS-based application. This is a richer setting compared to that of a traditional methodology, as one can envision different life cycles for serverless functions, microservices and data pipelines. The identified life cycles will be archived in both user documentation and online help to provide guidance on how to decide tool workflows and make the whole framework easy to use.

3.4 Quality assurance tools

A particular concern is raised by serverless FaaS for security and privacy. The increased granularity and expressiveness of RADON models will prompt the need to consider security and privacy with high priority in the architectural design. To deliver the business logic through serverless functions or microservices, developers have to make careful trade-offs in terms of non-functional requirements such as performance, costs, security and privacy. The decision of the optimal attack surface exposed by an architecture, as well as the implication of the fine-grained decomposition on non-functional requirements, ask for a rigorous methodology to help engineers select the right level of granularity.

RADON models will combine CDL annotations and generalized TOSCA models that support serverless functions, microservices and data pipelines. Developers will be able to analyze a RADON model using a hierarchy of logic programming and simulation techniques to determine whether the decomposition or aggregation of certain services produces an improvement in satisfying the specified requirements. This mechanism will make it possible to enact progressive decisions on the model, resulting in a solution with the optimal decomposition and the satisfaction of security and privacy policies in addition to usual non-functional requirements for performance and costs. Monitoring feedbacks will also be available on a dashboard for users to diagnose the runtime behavior of each application component and identify in a semi-automated manner what needs to be prioritized in the design.

The RADON quality assurance tools will be used to validate (i) IaC recipes, (ii) business logic encoded in serverless functions or microservices, (iii) data pipelines.

- For IaC recipes, a defect-prediction tool will be developed, combining anti-pattern detection and recent techniques from machine learning. It will address the intrinsic polyglot nature of infrastructure code, being either agnos-

tic of IaC technologies or specific to certain IaC defects and anti-patterns.
- For business logic, a continuous testing approach will be employed, through execution at the development stage immediately preceding the actual deployment of the application, to help detect unexpected issues before they are manifest in production.
- For data pipelines, users will be required to model data flows by customizing data generation profiles, which are needed to automatically produce test data with desired statistical characteristics, e.g., tailedness and burstiness, for verifying responsiveness and scalability annotated through the CDL.

4 Expected progress

RADON will touch upon various aspects of the state of the art, including microservices architectures, serverless FaaS, data technologies, topology and orchestration models, requirements engineering, continuous integration and continuous delivery as well as quality assurance. We describe in what follows the expected progress of our research agenda beyond the state of the art.

4.1 Microservices architectures

RADON will offer a holistic DevOps methodology to design, prototype, deploy, test, verify, and evolve microservices architectures. A variety of architectural patterns will be explored and encoded in the corresponding parametric TOSCA templates. The RADON framework will then enable the rapid prototyping of microservices architectures that combine microservices with serverless FaaS technologies, allowing a continuous spectrum of service granularities, a fine-grained capacity control through scaling at the function level, and deployment on resources and devices with heterogeneous capacities and at different locations (e.g., edge vs cloud backend). Several life-cycle management issues for microservices that are currently open will be tackled directly by the RADON methodology. These include, among others, devising strategies for granular monitoring, container-level anomaly detection, aligning performance regression testing with containerized microservices, finding appropriate modeling abstractions for microservices, and incorporating the container-based infrastructure in predictive models used to reason on optimal architectural decomposition. Holistic pattern-based design of microservices is a pending challenge and has the potential to radically improve the way in which microservices are reused, operated and evolved. In particular, there is a lack of methods and tools to establish the appropriate level of granularity or module decomposition while designing microservices architectures. In this respect, the scientific potential behind RADON is considerable: not only does RADON introduce a methodology for guiding software architects, and operators to understand and compose microservices, but also provide non-invasive, predictive tools to ensure the quality of microservice templates in terms of performance and correctness. Moreover, RADON advocates advancing along open standard and widely-adopted solutions like TOSCA which already have proven industrial and large-scale adoption, and may benefit from further experimentation and augmentation in the way of microservices and serverless computing.

4.2 Serverless FaaS

At present, the FaaS service model is not supported within model-based approaches to orchestration such as TOSCA. The challenge in extending these models to FaaS is considerable as one needs to move to modeling also the function dependencies of the application, retaining the ability to express dependencies and requirements on it, through triggers and events, as well as function parameters. The achievement of this extension will considerably broaden the expressiveness and applicability of model-based approaches to orchestration. Supporting tools to reason about the quality-of-service and cost of the resulting architecture are also required and present significant scientific challenges in their definition as one needs to tractably model a possible large number of triggers and actions, which is expected to result in state space explosion. These in turn require a characterization of the frequency at which certain triggers arrive, reasoning about performance risks (e.g., due to early dehydration of the container that servers the remote function), and an understanding of the different data caching policies that should be used when parts of the data is processed using FaaS services. Developing a methodology to reason and instantiate applications that encompass all these novelties thus represents an ambitious scientific endeavor.

4.3 Data technologies

RADON will adopt the data pipeline concept where the whole application is composed of independently deployable, schedulable and scalable pipeline tasks, regardless of whether they are microservices, serverless functions or self-contained applications. In comparison to other European projects, data processing will move from utilizing monolithic Big Data applications to adopting more general data pipelines consisting of freely composable, portable and reusable microservices for data transformation, storage and processing. RADON will also extend the TOSCA standard and its tools to support the design, deployment and automated life-cycle management (including programmatically establishing, configuring, scheduling, monitoring and destroying) of data pipelines composed of microservices and will support multiple data pipeline management frameworks (e.g., AWS Data Pipelines,[27] Apache Nifi[28]) and service mesh solutions.

4.4 Topology and orchestration models

RADON models will describe serverless FaaS-based solutions, which are currently unsupported in standardized orchestration languages such as TOSCA. Currently TOSCA mainly focuses on the description of the structure of the application to be deployed (topology templates) and imperative management processes (management plans). However, TOSCA does not support integrating behavioral information about the application itself in the deployment model, which is though required in FaaS since serverless virtualizes also the application logic. Thus, a significant extension to TOSCA developed in RADON will be the integration of behavioral aspects to enable the description of FaaS-based services.

4.5 Requirements engineering

RADON aims at defining a CDL to specify formal requirements and topology constraints and increase the automation in defining serverless-based applications. The CDL will use an abstract syntax and semantics that is natively close to orchestration abstractions and relationships available in TOSCA, thus easing bidirectional transformation, while also capturing application execution requirements, architectural properties, and security and privacy policies. RADON will couple the CDL with models that augment TOSCA and the Winery graphical environment for visual orchestration, offering a seamless way for the end user to recognize the model entities at play in the system, annotating CDL requirements that relate such entities, and obtain in return from the RADON framework recommendations on the optimal decompositions and generate mappings into concrete platforms and resources. The ability to annotate TOSCA models with requirements will allow to carry out automatically certain refactorings, for example by automating the addition of firewalls and other security-related services to meet regulatory and contractual constraints (e.g., legal interception by cloud providers that host applications defined using TOSCA).

4.6 Continuous integration and continuous delivery

RADON will extend capabilities of the existing tools by adding native support for orchestrating microservices, serverless FaaS and data pipelines, making them easier to manage, more reusable and portable. The tools will be further enriched with deployment templates that allow to reuse repeating architectural and technology patterns. RADON's advanced delivery modes will include canary testing schemes, dark launches, and A/B testing, which are generally available in commercial solutions but not in integrated open-source frameworks, a problem for the current portfolio of frameworks from European projects. The FaaS abstraction layer, centered around the notion of versatile event gateways acting as mediators for serverless calls, will provide a solution to proprietary lock-in that currently affects FaaS offerings, enabling security out-of-the-box and event-centric capabilities to traditional data sources.

4.7 Quality assurance

As current practices are not designed for the frequency of deployments in microservice- and FaaS-based systems, automation of the testing process is essential. In RADON, TOSCA will be extended to include testing annotations, which will provide both functional and non-functional test-related information. This will allow for automated setup of testing infrastructure. Input data for testing of built artifacts usually comes from fixed sample inputs, which provide debatable test coverage. In RADON, test inputs will also be made available from production data, which will be collected using available platform monitoring tools. It will include not only input data from users, but also extracted usage profiles, to simulate real user behavior in tests. When testing built artifacts that depend on other entities in the system, RADON will, again from production data, extract performance data of these other entities, to simplify test deployments. This means that instead of deploying complete application infrastructure, RADON will test artifacts with extracted simulations of entities they depend on. An additional layer of abstraction will allow for RADON to be able to use production data from different platforms, i.e., to achieve platform independence. RADON intends to pioneer IaC quality assurance, focusing in particular on developing tools for code smell detection and defect prediction for IaC. RADON tools will help improving the quality of IaC, assisting developers with machine-learning techniques to recognize code portions expected to be more prone to defects, based on training data from IaC libraries and codebases.

5 Related research projects

In this section, we review past and ongoing European projects where outputs are expected to feed or influence RADON.

[27] https://aws.amazon.com/datapipeline/.

[28] https://nifi.apache.org/.

A list of relevant projects is as follows:

– DICE[29] is a recently completed project that offers an IDE and DevOps tools for Java-based Big Data applications, leveraging on an extended UML profile and TOSCA.
– MODAClouds[30] offers an IDE, methodology and runtime framework to create and manage applications that run on multiple clouds. It is based on a model-driven engineering paradigm, rooted in the CloudML language, and integrated in the Modelio development environment.
– OpenReq[31] explores community-driven requirements engineering in the context of large and distributed software-intensive projects with tight inter-relations. The project aims at fostering a manageable continuous feedback stream from different types of stakeholders.
– STAMP[32] aims at pushing automation in DevOps one step further through innovative methods of test amplification, leveraging advanced research in automatic test generation. It reuses existing assets (test cases, API descriptions, dependency models), in order to generate more test cases and test configurations each time the application is updated.
– MIKELANGELO[33] focuses on making virtual infrastructures ready to run Big Data, HPC, and I/O intensive applications in production.
– WITDOM[34] aims at protecting the privacy and security of data outsourced to untrusted ICT providers, such as clouds, using cryptography and privacy-by-design.
– SEMIoTICS[35] aims at developing a framework for IoT application actuation. The approach covers security, privacy, interoperability, and dependability properties as well as embedded intelligence.
– CloudPerfect[36] offers tools for cloud providers to optimize the performance characteristics of their offerings through workload characterization, automated deployment and orchestration, monitoring and benchmarking.
– COLA[37] delivers a microservices platform (MICADO) along with runtime orchestration. COLA introduces its own domain-specific language similar to TOSCA YAML.
– ARCADIA[38] tackles challenges in design and deployment of highly distributed applications. The project introduces an execution model compatible with TOSCA NFV.
– SUPERSEDE[39] targets at providing advancements in end-user feedback and contextual-data analysis as well as DevOps-type decision-making. The major novel contribution is a new framework for instrumented software evolution and adaptation in the specific scope of data-intensive applications.
– CloudSocket[40] advances the research in multi-cloud computing, business process/service management, by offering a framework to manage BPaaS by considering all the necessary management levels involved and offering business-IT alignment. Context conceptual modeling in the context of the project is combined with machine intelligence techniques.
– CYCLONE[41] provides DevOps tools for multi-cloud management, covering virtual machine network accessibility, intra-site data access, inter-site data transfer, and resource scaling. Dimensions such as end-to-end security and federated network management are also taken into account.

6 Industrial use cases

Part of the challenge of investing in FaaS is to recognize the industrial use cases where the new technology is necessary to overcome technical and cost barriers. We illustrate three industrial use cases that demonstrate industrial domains in which the serverless FaaS paradigm can strike a contribution and there is a potential advantage in adopting a model-driven DevOps framework such as the one we have described. These use cases arise from the commercial needs of units within three European companies: Athens Technology Center (ATC), Engineering Ingegneria Informatica (ENG) and Praqma (PRQ).

6.1 Tourism promotion

ADAMO is an Android mobile application that combines the individual's tourist profile, a city's mobility network and points of interest (POIs) to share personalized experiences for travelers and residents, who can then discover the most exciting city routes customized to their preferences. The portfolio of news portals and touristic agencies will be able to suggest

29 http://www.dice-h2020.eu/.

30 http://www.modaclouds.eu/.

31 http://openreq.eu/.

32 https://www.stamp-project.eu/view/main/.

33 https://www.mikelangelo-project.eu/.

34 http://witdom.eu/.

35 https://www.semiotics-project.eu/.

36 http://cloudperfect.eu.

37 https://project-cola.eu/.

38 http://www.arcadia-framework.eu/.

39 https://www.supersede.eu/.

40 https://site.cloudsocket.eu/.

41 http://www.cyclone-project.eu/.

alternative routes for their clients and promote relevant data and services during route development. ADAMO may also play the role of a fair advertisement hub for local enterprises, while allowing city authorities to gain from the visibility of their profile. This will help showcase the touristic values of the city and connect preferable business offerings to stakeholders.

The current implementation of ADAMO is based on microservices and coarse-grained web services, which are developed in Java for the backend and in mobile Java for the Android app. The former hosts a set of components for collecting information about POIs in a city, analyzing data regarding the mobility and transportation support to move from one place to another and maintaining user personalization features. ATC exploits RESTful web services to coordinate communication among different components. These will be migrated to Docker[42] containers, applying the FIWARE[43] generic enabling (GE) technologies for (i) authentication and authorization (KeyRock, AuthZForce, Wilma); (ii) the management of POIs and routes (POI Data Provider, Object Storage, POI Proxy Swagger).

ADAMO is a general use case of RADON where novel data pipelines and data processing microservices, several FaaS-based, will be added to an existing architecture. With the help of the RADON framework, ATC will enrich ADAMO with a story-building environment, in which journalists will harness online contents collected from multiple sources (e.g., social media, blogs, news reports) to compose stories about POIs in a city. This development will facilitate the needs of news portals and tourist agencies for stories about specific routes. Specifically, data processing and AI technologies will be applied for the real-time analysis of online contents and their connection to POIs and routes. The resulting stories will be displayed on alternative routes to introduce POIs along them (Fig. 2).

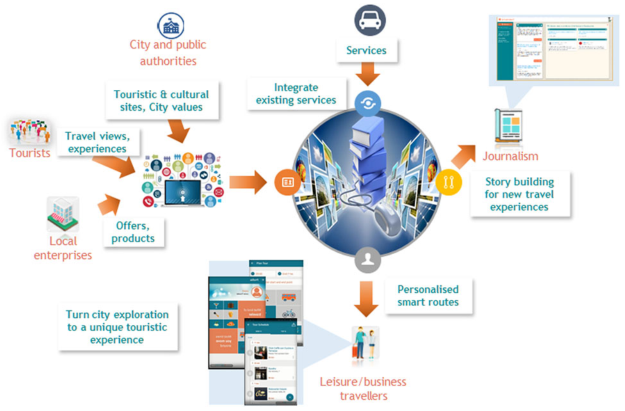
City and public authorities
Services
Tourists
Travel views, experiences
Touristic & cultural sites, City values
Integrate existing services
Journalism
Story building for new travel experiences
Local enterprises
Offers, products
Turn city exploration to a unique touristic experience
Personalised smart routes
Leisure/business travellers

Fig. 2 ADAMO: a use case for tourism promotion

6.2 Ambient assisted living

In recent years, eHealth technologies have been increasingly applied to assisted living in the home environment. ENG's use case falls within this application domain. It couples a horizontal platform, the CLOE-IoT middleware, on top of which runs a vertical application for ambient assisted living called AREAS SARA. CLOE-IoT is one of the ENG cloud offerings (CLOE), which aims to simplify the development of complex and robust IoT solutions across a wide variety of IoT devices and networks. SARA is an example of a CLOE-IoT distributed application envisioned to be part of AREAS, ENG's health ERP solution. It integrates robotic and IoT technologies to preserve quality of life for elders with mild cognitive impairment or Alzheimer's disease.

Built on the CLOE-IoT platform, SARA coordinates the following nodes available in the use case: Smart Phone, acting as the hub of a body area network of wearables for mobility detection and risk assessment; Robotic Rollator, a powered, wheeled walking frame, primarily used for physical support but also capable of identifying the patient, monitoring his mobility and navigating autonomously; Robotic Assistant, a robotic component connected to a (possibly dynamic) network of embedded devices and services for monitoring the patient's activities, health status and treatment/training progress as well as for supporting cognitive skills training and notifying the patient of upcoming treatments and visits; Environment Gateway, acting as the hub of smart devices embedded in the local physical environment and providing a variety of services ranging from appliance monitoring to patient tracking.

SARA is relevant to RADON as a prototypical use case where augmenting the capacity of the middleware can accelerate the redeployment, configuration and evolution of the vertical application consisting of robotic and IoT devices. Concretely, the use case will involve the use of the RADON model to capture the existing IoT/edge landscape. This will show in the process how to code new IaC recipes to set up the robotic and embedded devices at the edge, allowing ENG to develop a reusable library of deployment templates and configuration recipes for SARA. Serverless functions to capture events and triggering of actions will then be coded. Through integration of an open-source FaaS platform (e.g., OpenFaaS[44]) with CLOE-IoT, ENG will show how to increase the capacity of this industrial middleware to give SARA access to serverless computing technologies. Serverless functions will be then used to detect and react to specific events relatively to the environment (Fig. 3).

[42] https://www.docker.com.

[43] https://www.fiware.org.

[44] https://www.openfaas.com.

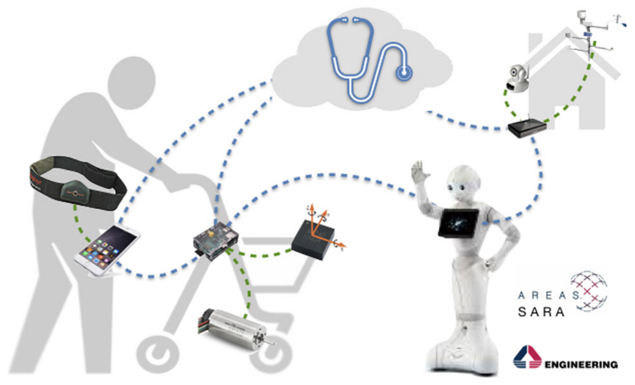

Fig. 3 SARA: a use case for ambient assisted living

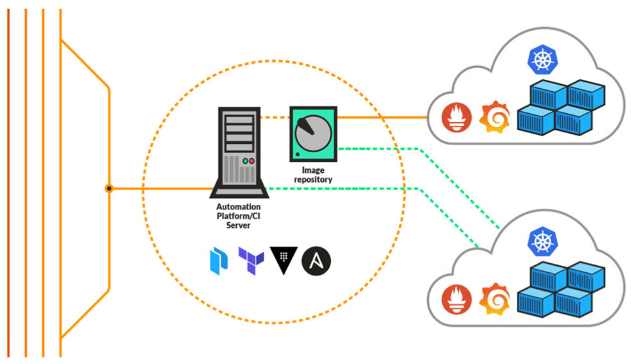

Fig. 4 Orbit: a use case for managed DevOps

6.3 Managed DevOps

Embracing DevOps in industrial setting requires accepting agile development and delivery principles throughout an organization, intersecting with the socio-technical systems that revolve around the software production process. PRQ is building a novel framework, called Orbit, that consolidates the company expertise in CI/CD methods and automation to implement DevOps-style governance in organizations. Managed DevOps, the focus of this use case, is conceptually similar to an outsourcing contract that a customer stipulates with an external company, like PRQ, to restructure the business with DevOps-style agile processes and adopt and maintain the related IT tools. This use case offers the opportunity to explore the interface between serverless, DevOps, socio-technical systems, and ensure that RADON itself is not perceived as a form of lock-in by adopters. As PRQ operates mainly in consulting, Orbit is envisioned as consolidating and expanding their market offering for digital transformation and will be offered to PRQ's customers across various markets, considerably enhancing the business potential of the company.

PRQ has noted a potential to further expand the use of FaaS through enrichment of DevOps governance with automatic FaaS-based triggers and actions that react to fine-grained events generated by developers, operators and all the human actors in the socio-technical system, while they work together in developing and delivering software. For example, a certain comment in the source code monitored by serverless functions may trigger a chain of actions to alert other developers in the team or be automatically picked up by the testing pipeline to reconfigure itself for an atypical test. Moreover, a large market has been identified towards integrating within Orbit a general-purpose function hub that consolidate the customer organization assets into libraries of functions that can be systematically reuse to rapidly create new business processes and IT-driven products. This will allow further reasoning on the function life cycle and long-term storage of technical and business process functions.

The primary goals of this use case are twofold: (i) to explore the adoption of DevOps through externally management services, which will be achieved through automation of IT actions needed to ensure that the different teams working on a software product can collaborate smoothly and efficiently, while following the correct setup of security and privacy rules; (ii) to explore the capabilities of the function hub component to archive complex process templates that are in use across industries relevant to this market segment, in order to enhance the capacity of the target organizations to start from day one to use serverless functions and build new processes on top of such stored business logic (Fig. 4).

7 Conclusion

A research agenda has been proposed for the RADON project, which aims at developing an innovative DevOps framework centered around serverless computing to unlock the advantages of the FaaS paradigm to industry. RADON will enable developers and operators to combine prepackaged functions and microservices into architectural topologies that are readily deployable, automating design, prototyping, deployment, testing, verification and evolution of FaaS-based applications under specified functional and non-functional requirements. A broad range of users can potentially benefit from the innovations foreseen by RADON as illustrated in our review of possible industrial use cases.

References

1. Baldini I, Castro P, Chang K, Cheng P, Fink S, Ishakian V, Mitchell N, Muthusamy V, Rabbah R, Slominski A et al (2017) Serverless computing: current trends and open problems. In: Chaudhary S, Somani G, Buyya R (eds) Research advances in cloud computing. Springer, Singapore, pp 1–20
2. Fox GC, Ishakian V, Muthusamy V, Slominski A (2017) Status of serverless computing and function-as-a-service (FaaS) in industry and research. arXiv preprint arXiv:1708.08028
3. Bass L, Clements P, Kazman R (2012) Software architecture in practice. Addison-Wesley, Boston
4. Gannon D, Barga R, Sundaresan N (2017) Cloud-native applications. IEEE Cloud Comput 4(5):16–21
5. Papazoglou MP, Vaquero LM (2016) Knowledge-intensive cloud services: transforming the cloud delivery stack. In: Kantola J, Karwowski W (eds) Knowledge service engineering handbook. CRC Press, Boca Raton, pp 472–517
6. Alrajeh D, Kramer J, Russo A, Uchitel S (2013) Elaborating requirements using model checking and inductive learning. IEEE Trans Softw Eng 39(3):361–383
7. Alrajeh D, Kramer J, Russo A, Uchitel S (2015) Automated support for diagnosis and repair. Commun ACM 58(2):65–72
8. Duvall PM, Matyas S, Glover A (2007) Continuous integration: improving software quality and reducing risk. Pearson Education, London
9. Humble J, Farley D (2010) Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, London
10. Jiang ZM, Hassan AE (2015) A survey on load testing of large-scale software systems. IEEE Trans Softw Eng 41(11):1091–1118
11. Leitner P, Bezemer C-P (2017) An exploratory study of the state of practice of performance testing in java-based open source projects. In: Proceedings of the 8th ACM/SPEC on international conference on performance engineering. ACM, pp 373–384
12. Heinrich R, van Hoorn A, Knoche H, Li F, Lwakatare LE, Pahl C, Schulte S, Wettinger J (2017) Performance engineering for microservices: research challenges and directions. In: Companion of the 8th ACM/SPEC on international conference on performance engineering. ACM, pp 223–226
13. Schulz H, Angerstein T, van Hoorn A (2018) Towards automating representative load testing in continuous software engineering. In: Companion of the 9th ACM/SPEC international conference on performance engineering. ACM, pp 123–126
14. Cois CA, Yankel J, Connell A (2014) Modern DevOps: optimizing software development through effective system interactions. In: Proceedings of the 2014 IEEE international professional communication conference. IEEE, pp 1–7
15. Shull F, Falessi D, Seaman C, Diep M, Layman L (2013) Technical debt: showing the way for better transfer of empirical results. In: Münch J, Schmid K (eds) Perspectives on the future of software engineering. Springer, Berlin, pp 179–190
16. Tufano M, Palomba F, Bavota G, Oliveto R, Di Penta M, De Lucia A, Poshyvanyk D (2015) When and why your code starts to smell bad. In: Proceedings of the 2015 IEEE/ACM international conference on software engineering, vol 1. IEEE, pp 403–414
17. D'Ambros M, Lanza M, Robbes R (2012) Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empir Softw Eng 17(4–5):531–577
18. Wurster M, Breitenbücher U, Képes K, Leymann F, Yussupov V (2018) Modeling and automated deployment of serverless applications using TOSCA. In: Proceedings of the 2018 IEEE international conference on service-oriented computing and applications. IEEE, pp 73–80