Migrating Smart City Applications to the Cloud

Michael Vögler and Johannes M. Schleicher
Technische Universität Wien

Christian Inzinger
University of Zurich

Schahram Dustdar
Technische Universität Wien

Rajiv Ranjan
Newcastle University

"Smart city" has emerged as an umbrella term for the pervasive implementation of information and communication technologies (ICT) designed to improve various areas of today's cities. Areas of focus include citizen well-being, infrastructure, industry, and government. Smart city applications operate in a dynamic environment with many stakeholders that not only provide data for applications, but can also contribute functionality or impose (possibly conflicting) requirements. Currently, the fundamental stakeholders in a smart city are energy and transportation providers, as well as government agencies, which offer large amounts of data about certain aspects (for example, public transportation) of a city and its citizens.

Increasingly, stakeholders deploy connected Internet of Things (IoT) devices that deliver large amounts of near-real-time data and can enact changes in the physical environment. Efficient management of these large volumes of data is challenging, especially since data gathered by IoT devices might have critical security and privacy requirements that must be honored at all times. Nevertheless, this presents a significant opportunity to closely integrate stakeholders and data from different domains to create new applications that can tackle the increasingly complex challenges of today's cities, such as autonomous traffic management, efficient building management, and emergency response systems.

Currently, smart city applications are usually deployed on premises. Cloud computing has matured to a point where practitioners are increasingly comfortable with migrating their existing smart city applications to the cloud to leverage its benefits (such as dynamic resource provisioning and cost savings). However, future smart city applications must also be able to operate across cities to create a global, interconnected system of systems for the future Internet of Cities.[1] Therefore, such applications have to be designed, implemented, and operated as cloud-native applications, allowing them to elastically respond to changes in request load, stakeholder requirements, and unexpected changes in the environment.

Here, we outline our recent work on the smart city operating system (SCOS), a central element of future smart city application ecosystems. The SCOS is designed to resemble a modern computer operating system, providing unified abstractions for underlying resources and management tasks, but

 2325-6095/16/$33.00

specifically tailored to city scale. We present the specific foundations of SCOS that enable a larger smart city application ecosystem,[2] allowing stakeholders and citizens to create applications within the smart city domain. This approach enables them to build applications by only focusing on their specific demand, while completely freeing them from the complexities and problems they're currently facing.

Challenges in Smart City Development

Applications in a smart city operate in highly dynamic environments comprising heterogeneous sets of infrastructures, which in turn are managed by different providers (transport, energy, water, and so on) to serve multiple stakeholders. To address the intrinsic complexities of such environments, we introduced a cloud-based smart city application ecosystem,[2] depicted in Figure 1. A number of challenges arise in the development of such an ecosystem.

The first challenge arises from incorporating and enabling the heterogeneous sets of available infrastructures in a smart city. On one hand, the ecosystem needs the ability to manage and operate the large number of devices that emerge through the IoT (sensors, gateways, and so on). This calls for novel means to stage, deploy, and organize such IoT devices to ensure that they can be fully utilized in smart city applications. On the other hand, smart city applications need to run on, and integrate, a plethora of IoT devices and cloud computing infrastructure types to operate holistically with maximum performance. To make this possible, the application ecosystem must be able to handle a variety of resources, ranging from traditional servers to hosted cloud solutions to the dormant computational potential of edge devices. Beyond this, the large number of different infrastructures and the rapid pace of infrastructure evolution call for means to move applications seamlessly between cloud, dedicated, and edge infrastructures in the smart city domain.

The second challenge comes from the massive amount of data that's emitted by a smart city in a large range of forms and formats. To enable data-driven applications, which are essential for decision making in smart city development, it's vital to provide intelligent data management mechanisms. These mechanisms need to provide adaptive abilities to store and manage heterogeneous data. Additionally, in a smart city context, they have to handle high-volume datastreams as well as large batches of data in structured and unstructured formats, which calls for novel integrated processing approaches.

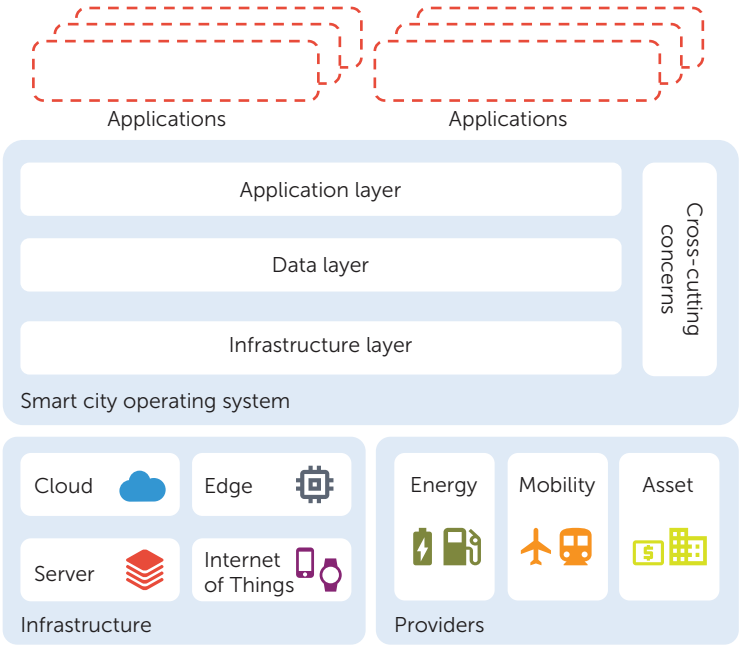

FIGURE 1. A cloud-based smart city application ecosystem. Designed around the central middleware—the smart city operating system (SCOS)—the smart city application ecosystem allows for the seamless integration of relevant stakeholders and resources to efficiently build, deploy, and operate smart city applications.

The third challenge arises from the requirements that come with developing, managing, and operating applications. Enabling smart city applications, which seamlessly incorporate data and infrastructure resources available in the ecosystem, requires both novel design and development approaches and novel architectural paradigms and patterns. To support practitioners in developing smart city applications on top of the domain's inherent complexities, it's vital to provide a toolset that hides these complexities, while also enabling practitioners to utilize and incorporate the aforementioned diverse infrastructure sets. This not only calls for novel methodologies and programming models, but also for the ability to provide an elaborate reconfigurable runtime environment that

enables adaptive execution along the whole range of heterogeneous infrastructures, multimodal data, and myriad IoT devices.

Finally, it's important to provide mechanisms to fully address the complex ownership and compliance requirements that arise in the smart city domain. An ecosystem in this domain has to deal with unique security and compliance constraints, which apply on many levels, ranging from company regulations to governmental restrictions on provider levels. Additionally, the heterogeneity of stakeholders, data providers, and data consumers, combined with the large scale as well as massive amounts of data, leads to an increased complexity.

To capacitate a smart city application ecosystem, addressing these challenges is essential, and in turn enables the execution of manageable and evolvable smart city applications.

Smart City Operating System

The SCOS, depicted in Figure 2, addresses the challenges we've described, providing a solid foundation for a larger smart city application ecosystem. To build a foundation that allows easy extendibility and high scalability, we followed the microservice architecture principle in its design.[3] This approach allows for clean separation of concerns and serves as a flexible interaction mechanism among SCOS components that enables novel synergies, leading to an extensible and evolvable system.

Infrastructure Layer

The first layer of the SCOS, the infrastructure layer, manages, configures, provisions, and constantly monitors the underlying infrastructure resources.

Infrastructure management. A smart city contains various heterogeneous types of infrastructure resources, such as traditional servers, cloud computing resources,[4] and edge and emerging IoT devices.[5] The infrastructure layer integrates these resources using an infrastructure management subsystem. The infrastructure management subsystem provides a mechanism that enables SCOS stakeholders to locate and identify their owned or leased resources. After the resources that should be managed have been located, they need to be accessible for the SCOS to be integrated into the overall system. Therefore, the infrastructure management subsystem provides a prebuilt list of drivers for communicating with the resources, as well as an access management component for securely storing the necessary credentials or keys. Finally, to provide an extensible approach, the infrastructure management subsystem provides APIs that enable stakeholders to build custom drivers to integrate emerging types of resources that require new forms of interactions.

Configuration management. Once the infrastructure resources are integrated into and accessible to the SCOS, it's necessary to facilitate their processing power, for example, to run and execute applications. To do this, configuration management allows for provisioning, deploying, and configuring these resources transparently and efficiently. Since the various types of infrastructure resources have different capabilities and environments, configuration management needs to respect these constraints. Hence, it provides a scalable and elastic provisioning solution that can be specifically tailored to each type of infrastructure.[6] The overall provisioning approach installs platform-specific software packages that allow leasing and releasing, monitoring, and deploying of resources in a generic and uniform manner. Next, for tailoring the connected and provisioned infrastructure to stakeholder requirements, the configuration management subsystem provides interfaces that enable the SCOS to configure several aspects (for example, pull- versus push-based updates) via the provisioned software packages. Finally, the configuration management subsystem supports the seamless deployment of single applications, complete application topologies, and additional necessary packages onto connected infrastructure resources by considering both the computational capabilities and the available execution environments.[7] Furthermore, since smart city applications need to be able to evolve over time to react to changing requirements or regulations, configuration management enables the seamless migration of application topologies among deployment targets.[8] Thus, it enables the independent evolution of applications, their topologies, and infrastructure resources.

Operations management. After the infrastructure is provisioned and ready for deployment, the SCOS

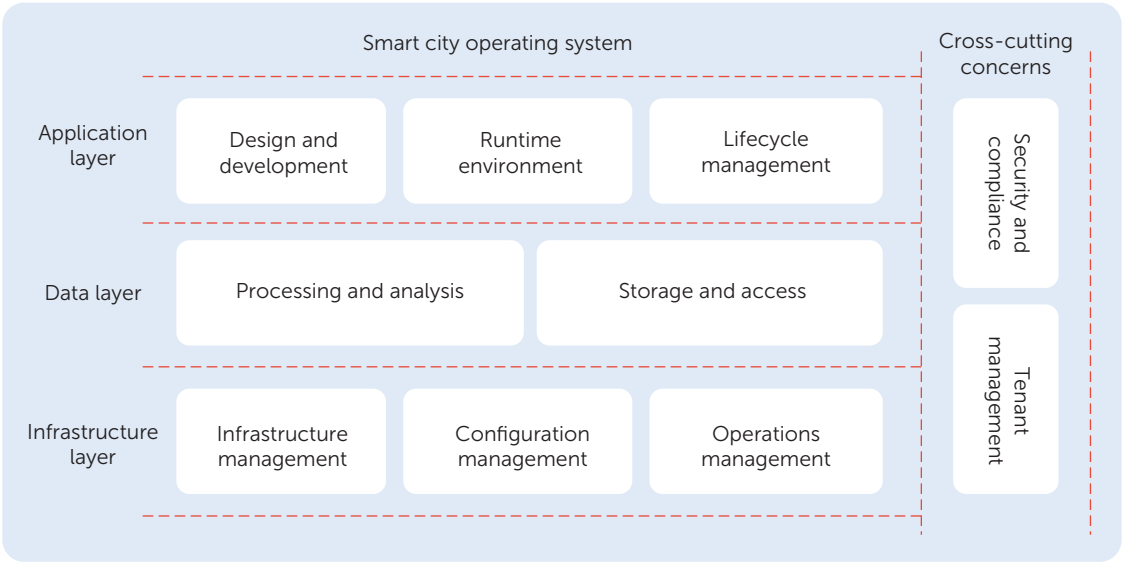

FIGURE 2. Smart city operating system. The SCOS resembles a modern computer operating system that is specifically tailored to the city scale by providing unified abstraction layers for underlying resources, management, and security aspects.

needs a mechanism for monitoring and analyzing the resources' performance. The operations management subsystem supports the constant monitoring and collection of information from connected resources by using the available monitoring capabilities of the respective infrastructure—for example, cloud monitoring APIs, such as Google's Monitoring API (https://cloud.google.com/monitoring/api) and commonly applied monitoring tools such as Ganglia[9]—or by provisioning software capabilities that support gathering performance measurements, such as tailored profilers for edge devices.[6] In addition to monitoring, operations management provides mechanisms to manage gathered logs, events, and faults. Based on collected information from the underlying infrastructure resources, the operations management subsystem conducts performance analyses that can be used to optimize resource utilization or evolve the overall infrastructure deployment. Furthermore, other SCOS subsystems can use the fine-grained analysis information to adapt application topologies to react to defined requirements such as service-level agreements (SLAs). Finally, operations management provides APIs that allow SCOS operators to define custom adaptation routines (such as scaling algorithms) to guarantee a defined availability for infrastructure resources or deal with network outages.[10]

Data Layer

The second layer of the SCOS, the data layer, is responsible for storing and providing access to data residing in the ecosystem, and processing and analyzing data that other SCOS layers can use to generate valuable insights.

Storage and access. In modern smart cities, running applications, infrastructure resources, and citizens produce an ever-growing amount of data (commonly referred to as big data[11]). The data layer provides the storage and access subsystem for managing and handling this data. This subsystem allows SCOS stakeholders to store and consume large sets of diverse data by providing generic and extendable APIs. For efficiently managing data in the SCOS, the subsystem supports the plethora of available data formats and the intrinsic diversity of data, and can also manage potentially noisy data[12] produced by the underlying infrastructure with its millions of managed resources. Additionally, the subsystem can

handle both static data that isn't frequently accessed or processed and dynamic data that constantly changes. To address the challenges that emerge from handling these different types of data, the storage and access subsystem provides a flexible approach that supports various storage facilities such as traditional relational databases, document-oriented databases, and complex unstructured datastores. Providing data storage in a data-as-a-service (DaaS) fashion enables the seamless integration of data facilities into the SCOS, eases the integration of new datastores, and allows for easy and uniform data access as well as storage functionality for components inside and applications on top of the SCOS. Furthermore, the storage and access subsystem provides mechanisms for merging and combining different types of data, which can be used as a foundation for analysis and planning operations in upper layers. Finally, the storage and access subsystem incorporates novel concepts that protect data, but also allow open data exchange where different levels of data owners can share data by integrating the hub of all things principle (http://hubofallthings.com). Following this approach enables the clear concept of ownership and supports emerging data compliance and security requirements.

One of the main goals of the SCOS application layer is to enable practitioners to create smart city applications in a simple, structured, and well-defined way.

Processing and analysis. After enabling the efficient access to and storage of data in the data layer, the processing and analysis subsystem allows stakeholders to efficiently transform, mediate, and analyze these large sets of diverse data. In addition, the processing and analysis subsystem provides an extensible set of established programming models for cloud, IoT, and edge resources, such as Apache Storm (http://storm.apache.org), Amazon IoT (https://aws.amazon.com/iot), Google Cloud Dataflow (https://cloud.google.com/dataflow), Amazon Elastic MapReduce (https://aws.amazon.com/elasticmapreduce), Apache Quarks (http://quarks.incubator.apache.org), and Esc,[13] that allows for processing both streaming and historical data, which is a vital aspect of current smart cities. On top of the basic programming model, the processing and analysis subsystem provides a novel processing approach based on a lambda architecture (http://lambda-architecture.net). Employing lambda architectures allows the SCOS to balance throughput, latency, and fault tolerance by simultaneously executing batch processing on batch data, and stream processing on real-time data. The analysis of processed data is another vital element in a smart city environment. Thus, the data layer provides data aggregation mechanisms, novel querying capabilities,[14] and a transparent environment for executing tailored processing and analysis routines. For building and deploying executable routines, the subsystem provides a prebuilt set of commonly applied processing and analysis logic, as well as a development kit for building and deploying custom code.

Application Layer

The third layer of the SCOS, the application layer, provides a comprehensive set of methodologies and tools for efficient design, development, distribution, and operation of smart city applications.

Design and development. One of the main goals of the SCOS application layer is to enable practitioners to create smart city applications in a simple, structured, and well-defined way. To accomplish this, the design and development subsystem is built around a comprehensive methodology for architecting, developing, and operating cloud-native smart city applications based on the methodology for architecture and deployment of cloud application topologies (MADCAT).[15] The methodology provides actionable guidelines for iteratively architecting and implementing smart city applications, both for new applications and for migrating existing applications to a cloud-native architecture[16] suitable for the SCOS

smart city application ecosystem. Furthermore, the SCOS provides a unified mechanism for describing smart city applications and their components along with deployment properties and requirements, similar to the Topology and Orchestration Specification for Cloud Applications (TOSCA).[17] This mechanism allows for easy sharing of applications and application components between SCOS deployments and provides a clear separation between application component dependencies and infrastructure requirements to create infrastructure-agnostic application descriptions. Such applications can then be offered in a smart city application market,[18] where users can buy and sell applications and their components using an open self-service platform.

Runtime environment. To allow for seamless execution of smart city applications, the SCOS runtime environment provides a configurable and adaptive execution environment for cloud-based applications that's independent of the underlying physical infrastructure. The execution environment incorporates a pluggable, unifying infrastructure abstraction[8] to transparently support and manage multiple application deployment mechanisms, such as container-based deployments—for example, Docker (https://docker.com)—and virtual machine-based deployments that are provisioned using predominant cloud offerings—for example, OpenStack (https://openstack.org) or Amazon EC2 (https://aws.amazon.com/ec2). The runtime environment furthermore provides a service mobility mechanism that allows for seamless migration of application components between datacenters and stakeholder premises.[19] Moving processing logic closer to data sources and/or data sinks can reduce network overhead and associated costs. Additionally, component migration allows for the execution of applications that otherwise couldn't be executed because of compliance constraints.

Lifecycle management. This subsystem is responsible for managing the complete lifecycle of smart city applications in the SCOS. For applications developed using the SCOS methodology we've described, the lifecycle management subsystem provisions required resources according to the specified requirements as well as applicable constraints, and deploys all application components according to their deployment manifests.[7,8] Stakeholders can then start, stop, or pause applications. During runtime, the lifecycle management subsystem will continuously monitor deployed smart city applications to optimize application deployment topologies.[20] Furthermore, monitoring data is pushed to the data layer, which enables consumption and further processing by applications. In addition to monitoring the overall operations of executed application components, the lifecycle management subsystem continuously monitors and verifies mandatory compliance criteria and enforces them by initiating component migrations if possible and instructing the runtime environment to deny access to critical resources if necessary.

Cross-Cutting Concerns

The final SCOS layer represents cross-cutting concerns. SCOS components or applications require common functionality (for example, authentication) that span several layers. Since such functionality affects the overall system, it's centralized in one place to avoid updating components throughout the system in case a certain behavior (such as logging) has to be changed.

Tenant management. Smart city applications operate under complex compliance and security regulations. Furthermore, since these applications must operate at large scale, are maintained by various stakeholders, and are provided in different possible facets, a plethora of constraints need to be efficiently managed. The tenant management subsystem supports the magnitude of participating stakeholders and allows them to specify their own security and compliance guidelines. An important aspect of tenant management is to enable the clean separation and isolation of any type of data, but especially sensitive data. Thus, tenant management enables each SCOS stakeholder to clearly define the following constraints regarding its data:

- Tenants specify which data they provide and in what quality.
- Tenants can decide what data can be shared or consumed.
- Tenants can describe with whom they want to share data, or who is specifically allowed to consume provided data.

- Tenants can specify what data and from whom they want to consume data.

Based on this specification, the tenant management subsystem derives a constraint matrix that clearly regulates data exchange in the SCOS, which prevents undesirable data transfer by respecting various forms of interactions (such as direct or transitive). In addition to data concerns, tenant management maintains a consolidated view of resources consumed by and available to SCOS tenants.

Security and compliance. Stakeholders in smart city environments implicitly expect and demand services to be secure, as well as to preserve their privacy. Thus, the SCOS provides a security and compliance subsystem that allows it to address both basic and complex security aspects:

- Since data in the SCOS constantly flows among different components or applications, which can reside inside or on top of the SCOS, the security and compliance subsystem provides strong encryption mechanisms to protect data in transit,[19] as well as approaches for securely storing sensitive data in SCOS components that deal with such data.
- Since the SCOS must deal with a broad variety of stakeholders and users, the security and compliance subsystem includes capabilities that facilitate strong authentication mechanisms that SCOS components can use to clearly specify who can access a specific service.
- The SCOS also provides authorization capabilities for enforcing permissions before accessing applications or manipulating data.
- To allow operators to manage the SCOS more efficiently, the security and compliance subsystem provides auditing and logging functionality at the component level.
- To keep the overall stack of components in the SCOS secure, the subsystem provides configuration management for automatically delivering software and security updates for different layers of the SCOS.

The security and compliance subsystem also deals with security requirements that emerge from the underlying infrastructure layer. Since IoT resources embody a vital aspect not only in enterprise systems, but also in consumer solutions, the security and compliance subsystem enables flexible security models. Based on these models, the SCOS can adapt to and respect emerging complex security requirements from the various domains it operates in.

With the rapid adoption of the smart city paradigm in cities around the globe and its respective success, more and more modern city capabilities are provided as smart city applications. This fact, combined with the plethora of supported ecosystems, diversity of stakeholders operating in the smart city ecosystem, and the magnitude of potential users, generates various challenges that need to be respected to build and provide truly future-proof smart city applications. The SCOS represents the key element for supporting ongoing smart cities as well as the foundation for enabling the future Internet of Cities.

References

1. J. Schleicher et al., "Towards the Internet of Cities: A Research Roadmap for Next-Generation Smart Cities," *Proc. ACM 1st Int'l Workshop Understanding the City with Urban Informatics*, 2015, pp. 3–6.
2. J. Schleicher et al., "Enabling a Smart City Application Ecosystem: Requirements and Architectural Aspects," *IEEE Internet Computing*, vol. 20, no. 2, 2016, pp. 58–65.
3. S. Newman, *Building Microservices*, O'Reilly Media, 2015.
4. M. Armbrust et al., "A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, 2010, pp. 50–58.
5. L. Da Xu et al., "Internet of Things in Industries: A Survey," *IEEE Trans. Industrial Informatics*, vol. 10, no. 4, 2014, pp. 2233–2243.
6. M. Vögler et al., "A Scalable Framework for Provisioning Large-Scale IoT Deployments," *ACM Trans. Internet Technology*, 2016, to appear.
7. M. Vögler et al., "DIANE: Dynamic IoT Application Deployment," *Proc. IEEE 4th Int'l Conf. Mobile Services*, 2015, pp. 298–305.
8. J. Schleicher et al., "Smart Fabric: An Infrastructure-Agnostic Artifact Topology Deployment

Framework," *Proc. IEEE 4th Int'l Conf. Mobile Services*, 2015, pp. 320–327.
9. M. Massie et al., "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing*, vol. 30, no. 7, 2004, pp. 817–840.
10. C. Inzinger et al., "Generic Event-Based Monitoring and Adaptation Methodology for Heterogeneous Distributed Systems," *Software: Practice and Experience*, vol. 44, no. 7, 2014, pp. 805–822.
11. C. Bizer et al., "The Meaningful Use of Big Data," *ACM SIGMOD Record*, vol. 40, no. 4, 2012, p. 56–60.
12. J. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things J.*, vol. 1, no. 1, 2014, pp. 3–9.
13. B. Satzger et al., "Esc: Towards an Elastic Stream Computing Platform for the Cloud," *Proc. IEEE Int'l Conf. Cloud Computing* (CLOUD), 2011, 348–355.
14. Q. Chen and M. Hsu, "Cut-and-Rewind: Extending Query Engine for Continuous Stream Analytics," *Trans. Large-Scale Data- and Knowledge-Centered Systems XXI,* LNCS 9260, Springer, 2015, pp. 94–114.
15. C. Inzinger et al., "MADCAT: A Methodology for Architecture and Deployment of Cloud Application Topologies," *Proc. IEEE Service Oriented System Eng.* (SOSE), 2014, pp. 13–22.
16. V. Andrikopoulos et al., "How to Adapt Applications for the Cloud Environment," *Computing*, vol. 95, no. 6, 2012, pp. 493–535.
17. T. Binz et al., "Portable Cloud Services Using TOSCA," *IEEE Internet Computing*, vol. 16, no. 3, 2012, pp. 80–85.
18. M. Vögler et al., "COLT Collaborative Delivery of Lightweight IoT Applications," *Proc. 2014 Int'l Conf. IoT as a Service* (IoTaaS), 2014, pp. 265–272.
19. J. Schleicher et al., "Nomads: Enabling Distributed Analytical Service Environments for the Smart City Domain," *Proc. IEEE Int'l Conf. Web Services* (ICWS), 2015, pp. 679–685.
20. M. Vögler et al., "Non-Intrusive Monitoring of Stream Processing Applications," *Proc. 10th IEEE Int'l Symp. Service-Oriented System Eng.* (SOSE), 2016, to appear.

MICHAEL VÖGLER *is a PhD student in the Distributed System Group at Technische Universität (TU) Wien, Austria. His research interests include cloud computing, service-oriented architectures, distributed systems, and the Internet of Things (IoT). Contact him at voegler@dsg.tuwien.ac.at or dsg.tuwien.ac.at.*

JOHANNES M. SCHLEICHER *is a PhD student in the Distributed System Group at Technische Universität (TU) Wien, Austria. His research interests include cloud computing, distributed systems, and smart cities. Contact him at schleicher@dsg.tuwien.ac.at or dsg.tuwien.ac.at.*

CHRISTIAN INZINGER *is a postdoctoral researcher in the software evolution and architecture lab (s.e.a.l.) at the University of Zurich. His main research focus is helping developers write better cloud applications, and his research interests include architectures for cloud applications, software evolution, and fault management in distributed elastic systems. Contact him at inzinger@ifi.uzh.ch.*

SCHAHRAM DUSTDAR *is a full professor of computer science heading the Distributed Systems Group at TU Wien, Austria. His work focuses on Internet technologies. He's an IEEE Fellow, a member of the Academy Europeana, and an ACM Distinguished Scientist. Contact him at dustdar@dsg.tuwien.ac.at or dsg.tuwien.ac.at.*

RAJIV RANJAN *is an associate professor (reader) in the School of Computing Science at Newcastle University, UK, and a visiting scientist at Data61, Australia. His research interests include cloud computing, content delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne. Contact him at raj.ranjan@ncl.ac.uk or http://rajivranjan.net.*