Enabling a Smart City Application Ecosystem

Requirements and Architectural Aspects

**Johannes M. Schleicher, Michael Vögler,
and Schahram Dustdar** • *TU Wien, Austria*
Christian Inzinger • *University of Zurich, Switzerland*

Cities are becoming increasingly "smart," but they need an underlying foundation that handles and manages all the lower-level complexities. In this article, the authors discuss the relevant challenges and building blocks of a Smart City Application Ecosystem (SCALE).

The recent advent and success of the smart city paradigm has led to its widespread adoption in cities and their supporting ecosystems around the globe. Spearheaded by leading international initiatives – such as the IBM Smarter Planet Initiative (www.ibm.com/smarter-planet), the Massachusetts Institute of Technology (MIT) Smart City Group (http://cities.media.mit.edu), Trinity College's Smart and Sustainable Cities Research (www.tcd.ie/research/themes/smart-sustainable-cities), as well as TU Wien's Urbanes Energie und Mobilitätssystem (URBEM; http://urbem.tuwien.ac.at) – more and more vital aspects of cities are becoming "smart." Various areas such as infrastructure, industry, and government (and, of course, citizens) of a smart city generate heaps of data and create sophisticated tangled interactions, leading to ever-increasing complexity.

Stakeholders in the smart city domain not only need to manage billions of sensors and devices emerging through the realms of the Internet of Things (IoT), but also need to be able to make informed decisions based on these massive amounts of data within these complex systems. This leads to numerous challenges[1] – ranging from the ability to manage and govern billions of devices, process and manage generated data, effectively and efficiently supply the resources to analyze them – up to matters of privacy and security. To manage this complex and continuously evolving system, here we introduce the concept of a smart city loop[2] that enables

stakeholders and citizens to manage this emerging complexity. Figure 1 shows this ideal smart city loop, a reactive system that addresses the aforementioned challenges on a conceptual level.

To enable this abstract concept and truly benefit from it, we need to create the foundation that handles and manages all the lower-level complexities. This should resemble a modern computer operating system by providing the same abstraction from low-level resources and management tasks, but at much larger scale and with additional abstractions. Thus, here we introduce the specific foundations of a Smart city Operating System (SOS) that enables a larger Smart City Application Ecosystem (SCALE). SCALE lets stakeholders and citizens create applications within the smart city domain, focusing on what they want to know and freeing them of most of the hassles and hindrances that this technology currently faces.

Approach

In the following, we outline the basic architecture of SCALE, introduce SOS (a core element in enabling SCALE), and outline challenges and requirements for smart city application engineering. (We discuss some additional aspects in the "Requirements" sidebar.)

SCALE's Architecture

Based on our experience with multiple research projects, industry collaborations, and smart city initiatives (such as URBEM), we present a high-level
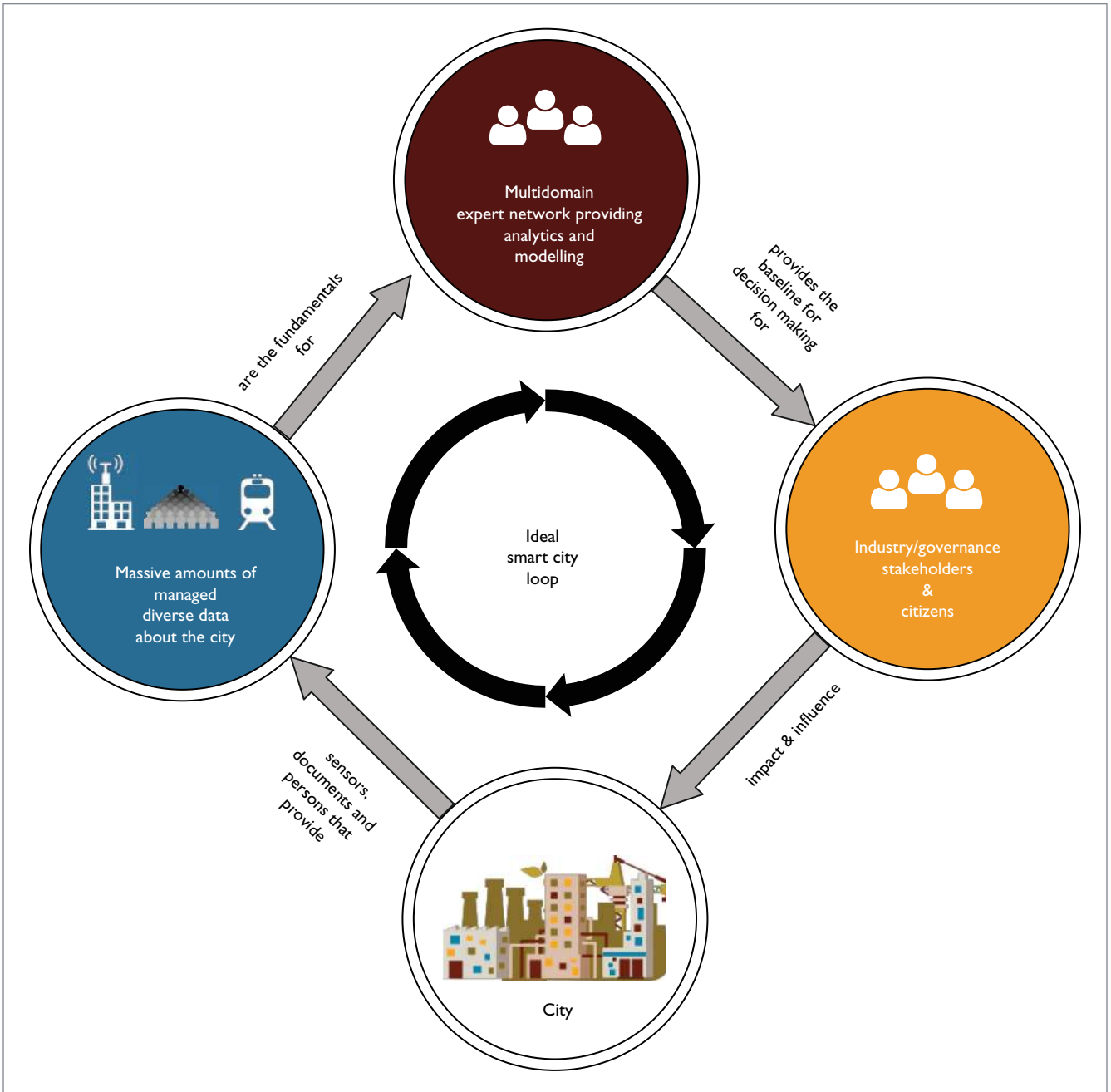
Figure 1. The ideal smart city loop. A conceptual model for a reactive system that addresses the challenges of today's smart city applications.

architecture of SCALE. (For a discussion of other projects and initiatives, see the related sidebar.)

SCALE is based on SOS, which addresses the aforementioned requirements. Naturally, smart city applications reside in a dynamic environment serving multiple stakeholders. Stakeholders do more than provide data for an application – they contribute functionality, or impose (possibly conflicting) requirements. So far, the fundamental stakeholders in a smart city are energy and transportation providers, as well as government agencies, which offer data about certain aspects (such as public transportation) of a city and its citizens. Traditional smart city applications are usually commissioned, operated by, and focused on single providers and their requirements,[3] leading to isolated vertical applications, and resulting in a significant fragmentation of smart city applications.

Requirements

To fully benefit from the novel possibilities brought by the advent of smart cities and the Internet of Things (IoT), stakeholders must be able to manage three major aspects. First, they need to operate the massive number of devices originating from the IoT domain. Furthermore, they need to deal with requirements that arise from such massive-scale infrastructures, starting from staging these devices to managing their evolution. Second, it's essential for stakeholders to analyze the ever-increasing amount of data, generated by billions of devices penetrating all aspects of a city's infrastructure. To achieve this, they need the necessary resources to perform analytics, as well as an efficient way to manage data and the analytics' results. Third, stakeholders need an effective way to plan based on this gathered information. These three aspects form the main pillars for the Smart City Loop (see Figure 1 in the main text), which is the vital baseline for stakeholders to stay on top of the massively emerging complexity within the smart city domain. However, to truly enable stakeholders, they must be able to build applications that let them focus on their areas of expertise without the burden of dealing with underlying complexities. It's therefore essential to pave the way for a Smart City Application Ecosystem that addresses these aspects.

To enable such an ecosystem, we must provide layers of abstraction to hide the complexities that come with operating, analyzing, and planning in the smart city domain. Similar to an operating system for traditional computers, an ecosystem must tackle the following areas.

First, it should provide the ability to manage and operate the massive amount of devices, as well as data providers in a smart city. This not only calls for means to stage, deploy, organize, and operate devices (to enable the Smart City Loop), but also requires abilities to adapt infrastructures based on novel insights in the form of smart city applications.

Second, it should enable data processing by allocating the necessary resources where and when they're needed. Due to the enormous heterogeneity emerging in the smart city domain, this calls for approaches that are able to handle a magnitude of different resources, ranging from traditional servers to hosted cloud solutions. Because these approaches also must be able to sustain infrastructure evolution, we need means to provide the baseline for decision making for sensors, documents, and persons to move applications between providers (that is, from a dedicated deployment to a hosted cloud and vice versa).

Third, to further enable the analytical models that provide the essential baseline for informed planning in the smart city domain, it's vital to provide smart data management mechanisms. The massive amount of data, combined with its intrinsic heterogeneity, calls for adaptive abilities to store and manage the data. Additionally, we need to handle high-volume data streams as well as large batches of data in structured and unstructured formats.

Finally, it's important to provide mechanisms to fully address the complex ownership and compliance elements that arise along with them. Especially in the smart city domain, we face several unique security and compliance constraints, which apply on many levels, ranging from company regulations to governmental restrictions on provider levels. Furthermore, the heterogeneity of stakeholders, data providers, and data consumers combined with the large scale and massive amounts of data, leads to increased complexity.

By introducing SCALE, we aim to break this traditional notion of industry verticals and according infrastructure silos to closely integrate requirements, functionality, and capabilities of multiple stakeholders in a seamless and manageable way. First, we strive to align smart city application engineering along the three pillars of methodology, modeling, and middleware.[2] Second, a comprehensive modeling framework will allow stakeholders to express distinct system capabilities in terms of declarative expectations and properties that allow for realistic simulations of real-world infrastructures, reducing the risks of developing against real systems, while at the same time increasing developer productivity. The simulation of real-world infrastructures will enable developers to plan ahead of time to develop and test applications for emerging, not-yet-available infrastructures. Third, SCALE will serve as a comprehensive communication, integration, and engineering tool throughout the complete application lifecycle, from requirements elicitation, definition, and specification, to development, deployment, and management of application functionality. Finally, by focusing on distinct, reusable units of functionality, SCALE allows for the creation of application components that can be composed, coordinated, and tested easily using a dynamic combination of simulated and real-world infrastructure. To achieve this, we'll provide a comprehensive middleware toolkit for engineering, executing, managing, and evolving smart city applications. This middleware toolkit will act as our SOS (see Figure 2).

Smart City Operating System (SOS)

To build an SOS on top of a future-proof architectural principle that allows for clean separation of concerns, easy extendability, and high scalability, we chose a microservice architecture. This architectural principle allows us to break out of

Related Work in Smart City Design

The smart city paradigm's rapid adoption and its broad coverage in research initiatives have led to several approaches, covering abstract concepts as well as specific tools to address emerging complexities.

SmartSantander[1] proposes a city-scale experimental research facility, supporting applications and services in the smart city context. SmartSantander focuses primarily on the Internet of Things (IoT) element of a smart city, and aims to provide a large-scale testbed that helps with issues arising from connecting and managing IoT infrastructure elements.

Jiong Jin and his colleagues[2] introduce an information framework for creating smart cities through the IoT. The authors focus on the urban information system as an omnibus volume, starting from the sensory level up to issues of data management and cloud-based integrations.

In a similar way, Nathalie Mitton and her colleagues[3] propose a combination of the cloud and sensors in a smart city environment. The focus of their work lies on the design of a pervasive infrastructure, where services interact with their surrounding environment with a clear focus on sensors in the IoT context.

In a more abstract conceptual direction, Hafedh Chourabi and his colleagues[4] present a framework to understand the concepts of smart cities. The authors provide a conceptual framework to comprehend the vital elements in a smart city by identifying critical factors. The framework suggests research directions and outlines practical implications.

Likewise, Taewoo Nam and Theresa Pardo[5] try to identify how a city can be considered smart, by aligning strategic principles in the context of technology, people, and institutions, which represent the main dimensions of a smart city.

References

1. L. Sanchez et al., "SmartSantander: IoT Experimentation over a Smart City Testbed," *Computer Networks*, vol. 61, Nov. 2014, pp. 217–238.
2. J. Jin et al., "An Information Framework for Creating a Smart City Through Internet of Things," *IEEE Internet of Things J.*, vol. 1, no. 2, 2014, pp. 112–121.
3. N. Mitton et al., "Combining Cloud and Sensors in a Smart City Environment," *EURASIP J. Wireless Comm. and Networking*, 2012, vol. 1, pp. 247–257.
4. H. Chourabi et al., "Understanding Smart Cities: An Integrative Framework," *Proc. 2012 45th Hawaii International Conf. System Sciences*, 2012, pp. 2289–2297.
5. T. Nam and T.A. Pardo, "Conceptualizing Smart City with Dimensions of Technology, People, and Institutions," *Proc. 12th Ann. Int'l Digital Government Research Conf. Digital Government Innovation in Challenging Times*, 2011, pp. 282–291.

---

traditional layered architectures, giving us the opportunity to have each component of SOS interact with any other, leading to novel synergies between SOS components. By enabling this flexible interaction model, we allow every component in SOS to benefit from any other component, leading to an evolvable and extensible system. In the following, we introduce the SOS's primary components and elaborate on their functionalities.

Infrastructure and resource management. The rise of the IoT and an ever-growing number of mobile entities has led to massive amounts of devices that provide data and need management. In the context of a smart city, these devices penetrate each level of a city's infrastructure, ranging from building management, to traffic control systems, and even to citizens. To deal with this complexity, the first layer of SOS provides the ability to manage and provision this heterogeneous infrastructure. This layer must enable efficient provisioning for both cloud and edge infrastructures, including the efficient management of infrastructure capabilities.

While at the first glance, traditional IT automation and configuration management tools such as Chef (www.chef.io/chef) or Puppet (https://puppetlabs.com) seem like a good fit, they unfortunately fall short in their ability to fully incorporate and manage edge devices such as the massive sensor arrays and gateways that emerge in the IoT domain. Furthermore, current IoT application-development methodologies consider the IoT devices as external dependencies that only act as data sources or command receivers. This, however, doesn't consider the available capabilities of the evolving edge infrastructure, even though they provide viable processing power. To incorporate these elements, we must rethink the role of edge devices and enable them as first-class elements in smart city infrastructures. To enable this, we need approaches like LEONORE,[4] a service-oriented infrastructure for elastically provisioning application components and capabilities on edge infrastructures at a large scale.

Next to edge infrastructures, the infrastructure and resource management layer also needs the ability to provision cloud resources on demand, based on application requirements. This calls for a way to incorporate multilevel elasticity control via a simple domain-specific language such as Simple-Yet-Beautiful Language (SYBL).[5]

Finally, we need to respect the ability to support infrastructure evolution, which is a vital and important part of the diverse smart city landscape. Therefore, the infrastructure and resource management layer needs to enable infrastructure-agnostic deployment
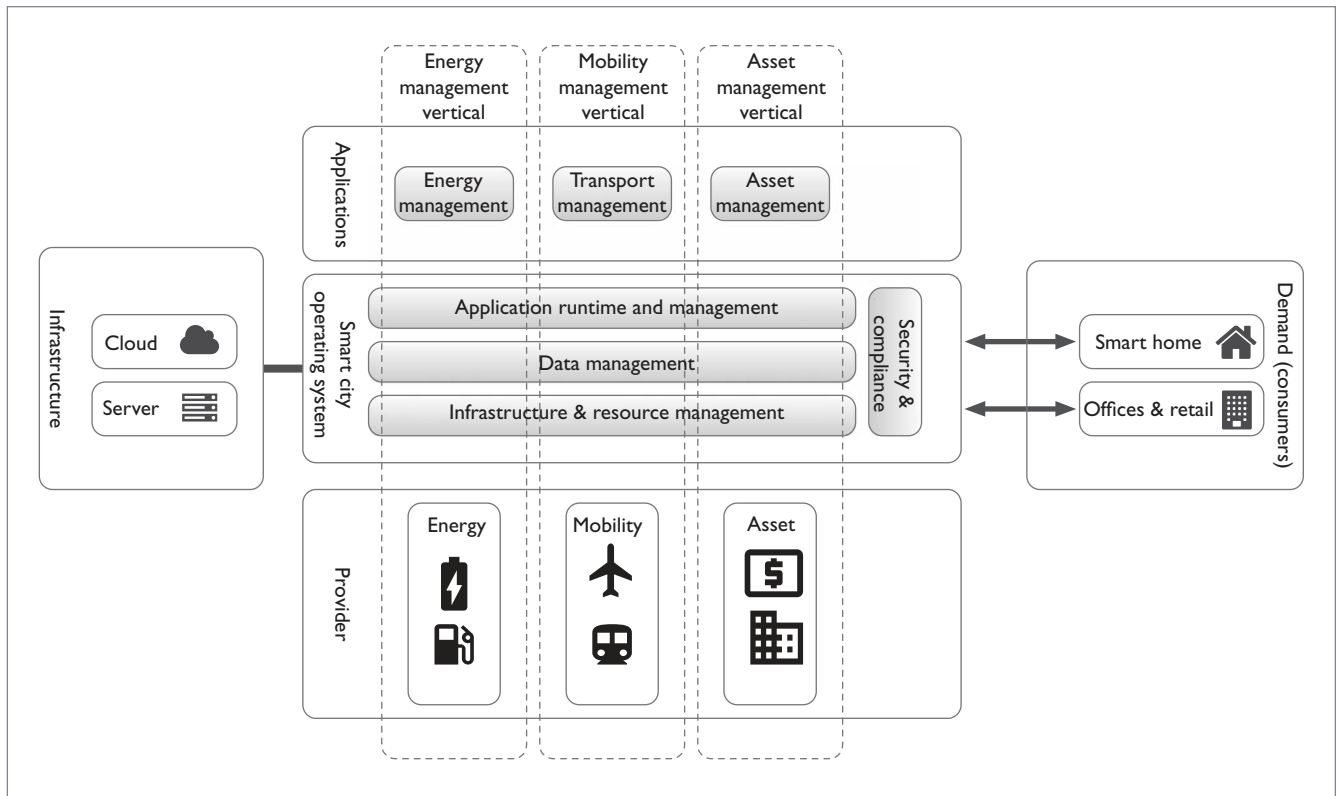
Figure 2. Architecture overview of the Smart City Application Ecosystem (SCALE). Designed around a central middleware — the Smart City Operating System — SCALE allows for the seamless integration of relevant stakeholders and resources to efficiently build, deploy, and operate smart city applications.

of artifact topologies that are able to respect the constraints of the required deployment infrastructure. This allows smart city applications to run on different stacks, ranging from on-premise server deployments to cloud solutions, which calls for an abstraction to model and describe the relevant entities in the smart city domain. The Methodology for Architecture and Deployment of Cloud Application Topologies (MADCAT)[6] provides a viable foundation as a methodology to describe application topologies independent of their deployment targets. Based on this methodology, Smart Fabric[7] presents a framework that supports the aforementioned infrastructure-agnostic deployment, thereby allowing infrastructures to evolve.

Data management. Modern smart cities' ever-increasing amount of data must be handled. To allow stakeholders to use these large sets of diverse data, commonly referred to as Big Data, the ability to efficiently manage, transform, mediate, store, and analyze is of utmost importance.

First, for managing data efficiently we need to respect the diversity of provided data, the plethora of different formats, and the fact that data generated by billions of devices will contain noise.[8] Next, data management in the smart city domain needs the ability to handle streaming data as well as historical data. These two types of data bring different challenges and require a flexible approach that allows merging and interweaving them as a baseline for further analytics and planning. Therefore, we must support a number of different storage facilities, ranging from traditional

relational data, to document-oriented and highly unstructured data stores. These different forms of data stores must be accessible in a data-as-a-service fashion, to allow effortless incorporation into the overall microservice architecture of SCALE, which enables other components or applications of SCALE to easily and uniformly consume data.

Because the rapid processing of data is another important aspect in the smart city domain, the data management component should support different stream-processing engines and approaches, like Esc.[9] Besides processing, data analysis is also a vital aspect of a smart city. Thus, we have to support advanced querying capabilities (as introduced by Qiming Chen and Meichun Hsu[10]), data aggregation mechanisms, and novel

approaches that combine data processing and storage, by employing lambda architectures. Furthermore, the data management component needs to incorporate novel concepts of data ownership. To enable new forms of open data exchange with full participation up to the citizen level, we should integrate principles such as the Hub-of-All-Things (HAT; http://hubofallthings.com), providing clear concepts of ownership, which enables an optimal integration for emerging security and compliance aspects.

Application runtime and management. Smart city applications are large-scale distributed systems that react and control, as well as analyze and reason about their physical environment, by using the underlying infrastructures. The inherently dynamic nature of smart city applications poses several challenges when running and managing such applications.

First, applications must quickly react on requirement changes. Second, applications have to deal with unreliable and expensive network links. Third, applications should provide a stable quality of service (QoS), even when experiencing infrastructure outages. Finally, applications need the ability to scale computations across different infrastructures, to handle the ever-increasing load generated by the smart city itself. Therefore, SOS provides the application runtime and management layer, which addresses these challenges.

By facilitating the previously described MADCAT methodology, SOS can separate application-specific topologies and requirements from the actual deployment infrastructure. Based on that, application components can be deployed on and run in different execution environments, ranging from traditional application servers running in the cloud, to more novel approaches that also consider other forms of infrastructure resources (for example, container-based environments provided by the edge infrastructure).

To deal with this plethora of available execution environments, SOS provides a generic and extensible approach for deploying and managing applications. Thus, the application runtime and management layer of SOS is built on top of the Dynamic IoT Application Deployment (DIANE),[11] which introduces a constraint-based model to represent the topology of an application and a mechanism to generate and provision infrastructure-optimized application deployments. According to that model, we can easily integrate with emerging execution environments in the IoT, such as Resin (https://resin.io), SmartThings (www.smarthings.com), and RIOT OS (www.riot-os.org), or support cloud-based IoT hubs.[12]

In addition to allowing seamless execution on different environments' respective infrastructures, SOS must consider that applications are running in a complex and dynamic context, which comprises available smart city resources as well as software configurations. Thus, the application runtime and management should provide optimal management of runtime resources and software configurations for applications. By acquiring resources via infrastructure and resource management, this allows applications to select required resources to meet service-level agreements (SLAs) and cost expectations. Finally, the uptake of this flexible application deployment and execution approach will help to create novel IoT application markets for storing and selling domain-specific application components. Therefore, COLT[13] provides a key pillar of SOS, by introducing a novel open market model that allows practitioners to collaboratively share, contribute, and manage the distribution of smart city application components.

Security and compliance. Applications in the smart city domain face complex security and compliance regulations. The large-scale nature of these applications, combined with the broad space of possible application types, leads to a huge amount of constraints that must be satisfied. Thus, the first aspect that must be considered is the plethora of participating stakeholders, where each has its own security guidelines and compliance specifications. Considering the magnitude of stakeholders, the real complexity, however, stems from the possible large-scale interactions that arise in this domain. Specific constraints that are valid for direct interactions might be violated in a transitive context, due to complex data regulations in these large-scale setups. Therefore, SOS addresses these emerging requirements by providing a security and compliance layer. To enable clean isolation and separation of sensitive data, tenant management is an essential element of this layer. Each tenant in SOS must be able to clearly specify which data it's able to provide, as well as define the provided quality, and finally choose which data can be consumed and by whom. The resulting specification enables us to derive a constraint matrix for exchanging data. Based on this constraint matrix, we avoid unwanted data exchange for both direct and transitive interactions, while still enabling the computation of data by utilizing capability migration approaches such as the one presented in Nomads.[14]

The second foundational aspect in this area emerges from the IoT domain. On the one hand, the sheer magnitude of devices calls for novel approaches in security management that have to deal with scalability requirements unknown to traditional server or mobile device domains. Furthermore, these devices are vital elements in every aspect of a smart city infrastructure, and are not only used in enterprise solutions, but also reach out into traditional consumer domains (such as smart meters and smart thermostats). Therefore, SOS provides security models that respect and adapt to these complex requirements

while still allowing stakeholders to enforce them by considering the currently constrained resources of devices in the IoT domain.

Smart-City Application Engineering

Future smart city applications must be designed, implemented, and operated as cloud-native applications that elastically respond to changes in request load, stakeholder requirements, and unexpected changes in their environment. The cloud computing[15] paradigm is a natural fit for smart city applications, because it allows engineers to create distributed applications based on dynamic resource allocation and elastic scaling. Developers can deploy scalable applications without the need for large, upfront infrastructure investments or significant operations expenses, paying only for resources that are actually consumed. Cloud computing offers many benefits for smart city applications, but novel software engineering methods must be investigated to allow for the efficient deployment and operation of geographically distributed applications that integrate and interact with large numbers of physical devices. We argue that future smart city applications must be designed as cloud-native, living applications that fulfill the following criteria, at a minimum.

Geographic awareness. Cloud providers typically provide datacenters that are geographically distributed across the globe. Cloud-native applications should be aware of which application components should be deployed in which datacenter, to minimize costs and maximize performance when interacting with data providers (for example, devices) and consumers.

Adaptation. Clouds are highly dynamic environments. Cloud-native applications must be able to react to changes quickly (for example, changes in customer patterns via elasticity,[16] changes in customer demands via continuous software delivery,[17] or changes in the cloud services landscape via cloud migration).

Resilience. Despite their advantages, cloud services are also known to be brittle (both in terms of performance and reliability). Cloud-native applications must be aware that cloud services could fail or degrade in performance without warning, and handle such situations gracefully.

A new generation of software engineering methodologies, architectures, and tools are required to enable the creation of cloud-native, living applications that are specifically designed to natively leverage a cloud infrastructure, as well as edge devices and infrastructure deployed throughout a city. Application components should be aware at all times of the environment they're executed in and be able to autonomously migrate between edge devices, datacenters, and cloud providers whenever necessary. Such a methodology will leverage the technical ideas of software-defined everything, microservice architectures, and infrastructure as code, combined with the scientific underpinning provided by agent-based and autonomic computing to enable autonomic and robust cloud applications.

Continuous monitoring and evaluation of their own state, including provided QoS, costs, requirements, and customer behavior, will allow collections of application components to react to changes and unforeseen events in their environment. Application component collections can autonomously modify their deployment topology by migrating all or parts of their components to different physical infrastructures (such as other datacenters or edge devices, where possible) to improve service levels or mitigate faults. Consequently, such component collections can also merge to consolidate services from multiple locations to one, to improve intercomponent network performance or to save costs. ▯

References

1. M. Naphade et al., "Smarter Cities and Their Innovation Challenges," *Computer*, vol. 44, no. 6, 2011, pp. 32–39.
2. J.M. Schleicher et al., "Towards the Internet of Cities: A Research Roadmap for Next-Generation Smart Cities," *Proc. 1st Int'l Workshop on Understanding the City with Urban Informatics*, 2015, pp. 3–6.
3. F. Li et al., "Web-Scale Service Delivery for Smart Cities," *IEEE Internet Computing*, vol. 17, no. 4, 2013, pp. 78–83.
4. M. Vögler et al., "LEONORE — Large-Scale Provisioning of Resource-Constrained IoT Deployments," *Proc. Service-Oriented System Eng.*, 2015, pp. 78–87.
5. G. Copil et al., "Multi-Level Elasticity Control of Cloud Services," *Service-Oriented Computing*, LNCS 8274, Springer, 2013, pp. 429–436.
6. C. Inzinger et al., "MADCAT — A Methodology for Architecture and Deployment of Cloud Application Topologies," *Proc. 8th Int'l Symp. Service-Oriented System Eng.*, 2014, pp. 13–22.
7. J.M. Schleicher et al., "Smart Fabric — An Infrastructure-Agnostic Artifact Topology Deployment Framework," *IEEE Int'l Conf. Mobile Services*, 2015, pp. 320–327.
8. J. Stankovic et al., "Research Directions for the Internet of Things," *IEEE Internet of Things J.*, vol. 1, no. 1, 2014, pp. 3–9.
9. B. Satzger et al., "Esc: Towards an Elastic Stream Computing Platform for the Cloud," *Proc. IEEE Int'l Conf. Cloud Computing*, 2011, pp. 348–355.
10. 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 Berlin Heidelberg, 2015, pp. 94–114.
11. M. Vögler et al., "DIANE – Dynamic IoT Application Deployment," *Proc. IEEE Int'l Conf. Mobile Services*, 2015, pp. 298–305.
12. R. Lea and M. Blackstock, "City Hub: A Cloud-Based IoT Platform for Smart Cities," *Proc. IEEE 6th Int'l Conf. Cloud Computing Technology and Science*, 2014, pp. 799–804.

13. M. Vögler et al., "COLT Collaborative Delivery of Lightweight IoT Applications," *Internet of Things. User-Centric IoT*, LNCS 150, Springer, 2015, pp. 265–272.

14. J. Schleicher et al., "Nomads-Enabling Distributed Analytical Service Environments for the Smart City Domain," *Proc. IEEE Int'l Conf. Web Services*, 2015, pp. 679–685.

15. M. Armbrust et al., "A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, 2010, pp. 50–58.

16. S. Dustdar et al., "Programming Directives for Elastic Computing," *IEEE Internet Computing*, vol. 16, no. 6, 2012, pp. 72–77.

17. D. Feitelson, E. Frachtenberg, and K. Beck, "Development and Deployment at Facebook," *IEEE Internet Computing*, vol. 17, no. 4, 2013, pp. 8–17.

Johannes M. Schleicher is a PhD student at the Distributed System Group at TU Wien, Austria. His research interests include cloud computing, distributed systems, and smart cities. Contact him at schleicher@dsg.tuwien.ac.at; dsg.tuwien.ac.at.

Michael Vögler is a PhD student at the Distributed System Group at 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; dsg.tuwien.ac.at.

Schahram Dustdar is a full professor of computer science (informatics) and he heads the Distributed Systems Group at TU Wien, Austria. His work focuses on Internet technologies. Dustdar is an IEEE Fellow, a member of the Academia Europaea, an ACM Distinguished Scientist, and recipient of the IBM Faculty Award 2012. Contact him at dustdar@dsg.tuwien.ac.at; dsg.tuwien.ac.at.

Christian Inzinger is a postdoctoral researcher at the software evolution and architecture lab (s.e.a.l.) at the University of Zurich. His main research focus is on 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; www.ifi.uzh.ch/seal/people/inzinger.html.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*