TOWARDS COLLABORATIVE SMART CITY MODELING

Dilshodbek Kuryazov, Christian Schönberg, Andreas Winter
Software Engineering Group, Carl von Ossietzky University of Oldenburg, Germany

Abstract. Recent technological advancements have led to an increasingly interconnected world, with new, autonomous "smart" devices and services that permeate everyday life (Internet of Things – IoT). This opens up a wide range of opportunities for new applications in diverse domains such as developing smart cities, smart houses, smart cars, smart grid, smart traffic control, etc. Simultaneously, these systems are reaching the new levels of complexity necessitating appropriate engineering methodologies. Model-Driven Engineering provides the required foundations for formally rigorous development of software-intensive systems.

Nowadays, model-driven approaches are extensively applied in designing, developing and maintaining software architectures, systems and platforms for smart cities i.e. smart city applications. As these smart city applications become huge and complex over time, collaborative work of several experts is required to cope with development and evolution challenges arising from smart city applications. This paper provides a comprehensive vision on applying model-driven collaborative development and maintenance, new upcoming trends in collaborative modeling of the IoT-based applications, and its adoption in developing smart city architectures.

1. Motivation

Technological advancement of recent years has brought up an increasing pervasiveness of the interconnected network of smart devices. More and more devices are being equipped with network connectivity to autonomously provide "smarter" services, forming the Internet of Things (IoT) [1]. Applications are wide-ranging, and have variously been termed "Smart X", including Smart Homes, Smart Cars, Smart Factories (Industry 4.0), Smart Government, Smart City, Smart Grid, Smart Traffic Control, and many more.

The concept of smart city arises from the need to manage, automate, optimize and explore all aspects of a city that could be supported and optimized by information technologies. The software paradigm IoT, being a core concept behind smart cities, is largely perceived as a collection of interconnected "things" within smart cities.

IoT-based smart city applications are realized by interconnected systems of heterogeneous hardware, software, and embedded systems: these cyber-physical systems introduce new levels of complexity, requiring appropriate engineering methodologies to support formally rigorous software systems and architectures development. Model-Driven Engineering (MDE) provides fitting foundations and is considered as an enabling technology for advancing Smart X applications. Smart City development is an extensive and complex endeavor, which requires intensive collaboration of various stakeholders and experts from different technical and social domains. Intensive collaboration by using model-driven approaches require broad support by appropriate tools and techniques. But, the current state of collaborative MDE is still a long way from realizing its full potential, and the adoption in industry and IoT remains limited.

A number of model-driven reference architectures [7] and models [14] are introduced for smart city development, so far. These architectures and models contemplate smart city architectures as a blueprint which provides an appropriate level of abstraction for the development process of smart cities. Model-driven reference architectures and models are used to represent and define different development aspects of smart city architectures gathering different views, viewpoints, software components, communication protocols, sensors, activators, etc. in a single, huge architecture. For instance, several model-driven approaches are investigated in [12] utilizing different modeling language profiles (e.g. standard UML profiles [3]) in development of smart city architectures. The standard UML profiles is also perfectly aligned in [15] for developing smart city architectures based on Internet of Things.

MDE intends to improve the productivity of the design and development, maintenance activities, and communication among various actors and stakeholders of a system. In MDE,

software models (e.g. in UML [3]), which also comprise source code, are the central artifacts. They are well-suited for designing, developing and producing large-scale software projects, architectures and applications for smart cities. Software models are the documentation and implementation of software systems being developed and evolved [2]. MDE brings several main benefits such as productivity boost and models become a single point of truth. Models are reusable and automatically kept up-to-date with the source code they represent [2].

Models are constantly changed during their development and evolutionary life-cycle by various developers and experts. They are constantly evolved and maintained, undergoing diverse changes such as extensions, corrections, optimization, adaptations and other improvements. All development and maintenance activities contribute to the evolution of software models resulting in several subsequent and parallel revisions. During software evolution, models become large and complex, raising a need for collaboration of several developers, designers and stakeholders (i.e. collaborators) on shared models and architectures i.e. *Collaborative MDE for Smart City*.

A meta-model generic, modeling tool generic and flexible collaborative modeling infrastructure was introduced in [4]. The associated technical support also focuses on providing a catalog of supplementary, operative services which are underlying services for building collaborative MDE tools for smart cities. This paper aims at providing vision and prospects to applying collaborative MDE infrastructure to developing smart city applications.

The remainder of this paper is structured as follows: Section 2 investigates existing model-driven software and system design and development approaches for smart city applications. The core concepts behind the proposed collaborative MDE infrastructure are explained in Section 3. Section 4 portrays the expected benefits of applying collaborative MDE infrastructure to smart city applications. This paper ends up in Section 5 by drawing some conclusions.

2. Model-Driven Engineering for Smart City: State of the Art

MDE has a much wider set of modeling techniques and a much more detailed separation of different views and concerns [7]. For instance, components have well-defined interfaces and ports. MDE benefits from its capabilities to model different, but integrated views and behavior of systems which are eventually made executable for different platforms. The basic idea of the smart city vision is the pervasive presence of a variety of things or objects such as sensors, actuators, mobile phones, etc. which are able to interact with each other and cooperate with their pairs to reach common goals [1][5]. This section shortly reviews MDE approaches and methodologies for developing smart city applications that the collaborative MDE infrastructure can be applied to.

There are several MDE approaches for developing smart city applications, e.g. the Sirius-based ThingML language [6]. These model-driven approaches provide very expressive modeling language of systems, possibly with source code generation. As long as the main motivation for MDE is to describe a system on a higher level of abstraction, this is usually done in UML and other languages by diagrams modeling specific aspects or views of a system.

UML designer [9] is another domain-specific modeling framework built on the top of Sirius and EMF. In UML designer, smart city architectures and models may describe its logical role of classes by *class diagrams*, system-actor interactions by *use case diagrams*, architectural models by *component diagrams* as well as *deployment diagrams*, which show the mapping of components to physical entities. Behaviors can be described by *sequence diagrams*, by *state machines* and *activity diagrams*. Activity diagrams describe the actions and object and control flows between actions. State machine diagrams are used in several embedded domains to model the behavior of specific objects e.g. the discrete behavior of components, in a model-driven environment, is usually defined through finite state machines.

The research presented in [12] takes advantage of the MDE principles to build a holistic development methodology involving a common, semantically expressive abstraction model, to specify a smart space with its specific services. It proposes the Resource-Oriented and Ontology-Driven Development (ROOD) methodology, which improves traditional MDE-based tools through semantic technologies for rapid prototyping of smart spaces according to the IoT paradigm. In the

framework of ROOD, the Smart Space Modeling Language (SsML) was developed based on UML, that defines a Domain Specific Model (DSL). It can be used for describing high-level behaviors, interactions and context information of the entire smart space. It further defines the processing aspects related to the sensing and actuating capabilities of the smart objects, as well as the context model they manage; moreover, encapsulate these concepts into RESTful resources. The ROOD approach is realized using Obeo Designer [13].

Lessons Learned. As explained above, there are already several domain-specific MDE notations and tools that can be reused for designing, modeling and developing smart city architectures and applications by distinguishing different design aspects and perspectives such as views, viewpoints, components, communication protocols, etc. For the sake of interactivity by collaborative development, maintenance and consistency, evolution and flexibility, they mostly lack collaborative designing and development support by sharing the artifacts of smart city architectures, models and applications among collaborators. The most existing tools use standard UML profiles by defining their own notations. However, these tools are developed on the top of EMF – eclipse modeling framework, Sirius and Obeo Designer using Ecore-based meta-modeling feature and standard UML meta-models. The collaborative modeling infrastructure introduced in Section 3 will be applied to these existing tools using their underlying meta-models.

3. Collaborative Modeling Approach

A meta-model and modeling tool generic as well as flexible collaborative MDE infrastructure was introduced in [4]. The overall approach consists of a three-layer architecture namely: *language generation, service orchestration* and *applications*. In order to use the collaborative MDE infrastructure in any modeling domain, a *difference language* has to be generated for each domain. After generating difference languages, other underlying services can be orchestrated for building collaborative MDE applications for each domain language. These techniques are also applicable to the collaborative MDE of smart city applications.

Language Generation. The collaborative modeling approach takes advantage of *modeling deltas* [4] as difference documents for storing and exchanging model changes among collaborators. Model changes in modeling deltas are represented by a textual, operation-based *Difference Language (DL).* Formally, DL is a family of domain-specific languages. Specific DLs for domain-specific modeling languages can be generated by *DL Generator,* importing the meta-models of modeling languages. For instance, the approach is applied to UML class diagrams by importing the UML class diagram meta-model in [4].

As identified in Section 2, the most existing MDE tools are developed on the top of EMF – eclipse modeling framework using Ecore-based meta-modeling feature. DL will be applied to the existing MDE tools (e.g. currently, to UML designer, to ThingML which is an EMF- and Sirius-based [10] domain-specific modeling tool [9] and to SsML in future). Specific DSL will be derived from the EMF-based Ecore meta-model [8] based on the standard UML profiles.

Service Orchestration. In order to embed the collaborative MDE support behind the existing MDE tools for smart city architectures, there is a need for several operational services to perform certain collaborative modeling operations. These operations might, for instance, be calculating modeling deltas by listening for changes or comparing subsequent revisions, applying or propagating modeling deltas on models, etc. The collaborative MDE infrastructure [4] further provides a catalog of supplementary services that can recognize the DL syntax as well as manipulate and reuse DL-based modeling deltas. After generating a specific DL for a domain-specific modeling language, these services can be orchestrated in order to perform certain operations in collaborative MDE of smart city architectures. In the following, these services and their orchestrations are utilized in collaborative modeling support for smart city architectures.

Applications. In the collaborative development of smart city architectures, the overall infrastructure of collaborative MDE has to provide two main scenarios: (1) *interactivity* of collaborators; (2) *consistency* of centralized artifact repositories. *On the one hand,* providing *interactivity* among collaborators is a main concern. Collaborators may use domain-specific MDE

tools for working on their local workspaces. But, there must be a support for joining/opening centralized and shared smart city architectures and working on it in parallel with other collaborators. Simultaneously, the changes, they make on their copies of model, should be synchronized with other parallel instances in real-time, which is referred as *interactivity*. *On the other hand,* smart city architectures and models have to be stored in the centralized and persistent repositories. It allows for storing the histories of architectures and models under development and evolution. Repositories can then store and persist models and their histories safely for further reuse and manipulations.

The collaborative MDE infrastructure [4] introduces two main scenarios of collaborative MDE namely *micro-versioning* and *macro-versioning.* Micro-versioning together with domain-specific MDE tools enables *interactivity* of several designers and developers (by concurrent collaboration) on the shared and centralized model-driven smart city architectures. Macro-versioning provides the *consistency* of model-driven smart city architectures. It is a centralized modeling delta repository with model management features such as opening working copies of models, reverting their revisions, storing complete model and their revisions, etc.

4. Expected Benefits

The central theme in this paper is to contribute towards a needs-based improvement of collaborative MDE techniques, methods, and tools, and researching novel, smart modeling techniques and applications, to address challenges posed by future software-intensive systems in the framework of smart city architectures and models. Some of the most important challenges include the following:

Interactivity: The interconnected world enables global software engineering, with developer teams being distributed over long distances already becoming a common practice. For incremental and agile model-driven development and engineering, this highlights a need for truly model-driven collaborative work, and tools for collaborative development of smart city architectures. The *micro-versioning* is to be considered as a main foundation for providing interactivity support, treating a centralized model as a single point of truth. Synchronization of changes will be eased by exchanging small DL-based modeling deltas. The various domain-specific MDE tools (e.g. UML Designer, ThingML, SsML, etc.) which are running on different platforms can communicate with each other in terms of DL.

Consistency: Both the heterogeneity and the high complexity of cyber-physical systems make integrated views and models of diverse development artifacts and their interrelationships indispensable. Model consistency and integration needs to occur both throughout evolutionary life-cycle of the systems/models under development. With collaborative modeling, the histories of evolving smart city architectures are consistently preserved in modeling delta repositories by *macro-versioning* for further reuse and analysis. The DL syntax fully satisfies MDE concepts and provides consistency of modeling deltas as well as models under development and evolution.

Flexibility: Smart city applications are dynamic, fast-evolving, and based on heterogeneous parts, e.g. realized as micro-services. This raises a need for technology-agnostic approaches to integrate diverse subsystems, flexibly. MDE can be applied for model-driven systems integration, bridging the gap between service and implementation layers using model transformations, while preserving separation of concerns. DL does not rely on any specific underlying implementation or technical space being fully independent from underlying problem domain. The DL-based modeling deltas may form MDE artifacts confirming to a given meta-model, that can represent complete systems and their changes. The collaborative MDE infrastructure [4] fully provide the suitable model-driven collaborative development for smart city applications.

5. Conclusions and Future Work

This paper has demonstrated a promising vision and prospects to applying collaborative MDE infrastructure [4] to engineering and developing the smart city applications. As long as there are already sufficient domain-specific MDE and development tools for smart cities, development of

such tools is out of the scope, here. But these approaches lacking support for collaboration, which will be provided as an add-on to the given tools, by the techniques presented in that paper.

The collaborative MDE infrastructure was already successfully applied to designing and developing UML class diagrams and successfully evaluated by the teams of Uzbek and German collaborators (with the server and collaborators located at the University of Oldenburg and collaborators located at Urgench branch of Tashkent University of Information Technologies). The tool can be downloaded and tested at [16]. Currently, collaborative MDE infrastructure explained in Section 3 is being applied to UML designer and, in future, to ThingML which is an EMF- and Sirius-based [10] domain-specific modeling tool [9] as well as SsML – Smart space Modeling Language.

References

[1] L. Atzori, A. Iera and G. Morabito: *The internet of things: A survey.* Computer networks, 54(15), pp. 2787-2805. y. 2010.

[2] A. Kleppe, J. Warmer and W. Bast: *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 2003.

[3] J. Rumbaugh, I. Jacobson and G. Booch: *Unified Modeling Language (UML) Reference Manual.* Pearson Higher Education, 2004.

[4] D. Kuryazov, A. Winter and R. Reussner: *Collaborative Modeling Enabled by Version Control*, In: I. Schaefer, D. Karagiannis, A. Vogelsang (eds): Modellierung 2018, Lecture Notes in Informatics (LNI), vol. P-280, pp. 183-198, Bonn, Gesellschaft für Informatik (GI), Feb. 2018.

[5] D. Giusto, A. Iera, G. Morabito and L. Atzori (Eds.): *The Internet of Things.* Springer, 2010.

[6] F. Franck, et al.: *MDE to manage communications with and between resource-constrained systems.* MDE Languages and Systems. Springer Berlin Heidelberg, pp. 349-363, 2011.

[7] Kateule, Ruthbetha; Winter, Andreas: *Architectural Design of Sensor based Environmental Information Systems for Maintainability*, Springer, Magdeburg, 2018.

[8] D. Steinberg, F. Budinsky, E. Merks, M. Paternostro: *EMF: Eclipse Modeling Framework*. Addison-Wesley Longman Publishing Co., Inc., 2008.

[9] Obeo Network: *UML designer*. http://www.umldesigner.org/, visited on 02.10.2017.

[10] V. Viyović, M. Maksimović, and B. Perisić: *Sirius: A rapid development of DSM graphical editor.* 18th International Conference on Intelligent Engineering Systems (INES), IEEE, pp. 233-238. July, 2014.

[11] F. Fleurey, M. Brice, and S. Arnor: *A model-driven approach to develop adaptive firmwares.* In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 168-177. ACM, 2011.

[12] I. Corredor, A. Bernardos, J. Iglesias, J.Casar: *Model-driven methodology for rapid deployment of smart spaces based on resource-oriented architectures.* Sensors, 12(7), pp.9286-9335. 2012.

[13] Obeo Designer, 2016. Domain Specific Modeling for Software Architects.

[14] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David: *A literature survey on smart cities*. Science China Information Sciences, *58*(10), pp. 1-18, 2015.

[15] K. Thramboulidis and F. Christoulakis: *UML4IoT—A UML-based approach to exploit IoT in cyber-physical manufacturing systems*. Computers in Industry, *82*, pp. 259-272, 2016.

[16] Project Group: Real-time collaborative modeling tool for UML class diagrams, Software Engineering Group, Carl von Ossietzky University of Oldenburg, Germany, 2014, URL: https://pg-kotelett.informatik.uni-oldenburg.de:8443/build/stable/