smart cities

MDPI

Article

Smart City Solution Engineering

Jerker Delsing

Embedded Intelligent Systems Lab., Lulå Univeristy of Technology, 97187 Luleå, Sweden; jerker.delsing@ltu.se; Tel.: +46-70-626-1931

Abstract: Many smart city applications have been proposed and demonstrated over the years; however, moving to large-scale deployment is still scarce. A contributing factor to this scarcity is the lack of well-established engineering methodologies for large-scale smart city applications. This paper addresses engineering methodologies and tools for large-scale smart city application engineering, implementation, deployment, and evolution. A model-based engineering approach based on IoT, SOA, and SysML is proposed and applied to a smart streetlight application. Engineering considerations for streetlight area enlargement and updated technology generations with additional capabilities are discussed. The proposed model-based engineering approach provides considerable scaling simplifications and opportunities for considerable savings on engineering costs. The model-based engineering approach also provides good documentation that enables technology evolution specifications that support both maintenance and emerging behaviours.

Keywords: engineering; modelling; SOA; Arrowhead

Citation: Delsing, J. Smart City Solution Engineering. *Smart Cities* **2021**, *4*, 643–661. https://doi.org/10.3390/smartcities4020033

Academic Editor: Pierluigi Siano

Received: 24 March 2021
Accepted: 27 April 2021
Published: 30 April 2021

1. Introduction

Smart cities have been discussed for quite some time in the scientific community, industries, and societal groups, and a number of reviews on the topic are available. There are properties related to smart cities that distinguish them from other domains, such as Industry 4.0 production, smart energy grids, and smart houses. Such distinguishing properties are as follows:

- Very large number of devices and users >10^5;
- Very large number of independent functionalities with emerging behaviours and interactions;
- Extremely heterogeneous technologies; and
- Extremely distributed environments >10 km^2.

Moving the smart city ideas and experiments into real-world realisation, while enabling the above properties, is a current challenge. The further enabling of widespread smart city functionalities and their integration will require economically and commercially sound implementations. This will put new demands on the engineering, deployment, operation, maintenance, and evolution of smart cities.

Such demands can be transferred into a number of more detailed requirements that need to be addressed. Such requirements are, e.g., as follows:

- Architecture for smart cities;
- Engineering of smart city solutions;
- Operation of smart city solutions;
- Maintenance of smart city solutions;
- Evolution of smart city solutions;
- Training related to smart city engineering, operations, maintenance, and evolution;
- Smart city security; and
- Smart city safety.

Clearly, the details of these requirements will be viewed differently depending on the "smart" city functionality. The enabling of integration between functionalities and their evolution will call for architectures allowing for run-time engineering, management, maintenance, and evolution. It is clear that smart city functionalities have to be addressed using some type of system of systems (SoS) thought and architecture. Fundamental properties of SoS can be found in [1,2].

Current technology trends point towards the usage of service-oriented architecture (SOA) and microservices, as a well-accepted approach. An interesting example of a detailed implementation architecture based on microservices is [3]. The engineering of SoS SOA solutions has been discussed since early 2000. SOA is, to a large extent, the dominant approach for creating automation and digitalization solutions in industrial production, smart grids, smart environments, etc. We currently find a smaller number of edge solution frameworks/platforms; a recent review of such industrial frameworks/platforms is [4]. However, in what way can these types of platforms be used for smart city engineering? The engineering of SOA/microservice-based solutions has some history in the area of business computing (e.g., banking) and cloud-based computing [5]. Moving SOA and microservice solutions to the edge has been a focus area of multiple European projects over the last decade. Important publications demonstrating this development are, e.g., Colombo et al. [6,7] and Delsing [8,9]. For the engineering of SOA-based solutions, Urgese et al. [10] recently published an approach for modelling and integrating a solution engineering process using an SOA approach. This approach allows for modelling and integrating various solutions at different stages of engineering, operations, and evolution.

This approach to smart city functionality and integration engineering is model-based. Service-oriented modelling has been addressed since early 2000. A prominent example is [11].

The smart city engineering approach applied herein is based on an SOA/microservice architecture. Thus, we need to consider an SoS SOA architecture. For the modelling thereof, SysML is used. As an SoS SOA architecture backbone, the Eclipse Arrowhead reference architecture and implementation platforms have been used [12].

The intention is to show how model-based engineering can serve in design and engineering of complex smart city solutions where advanced and dynamic boundary operational conditions are at hand. Thus, the focus of this paper is on how simple functionalities can be engineered to integrate into more complex functionalities in a smart city context.

In summary, this paper focusses on smart city engineering, both in design and run-time.

2. Related Work

The knowledge field of smart cities is very broad. Given the geographical size of smart cities, it is obvious that updates and extensions (e.g., geographically) of single functional solutions and installation of new functionalities have to occur in parallel to the operation of existing functionality solutions. Integration between existing, upgraded, extended, and new functionalities is one area that has a large dependence on interoperability, integrability, and composability of the involved functionalities. Thus, the engineering of smart cities has to consider not only design time engineering, but also maintenance of the smart city, its evolution, and emerging behaviours.

Current literature on smart city engineering is limited. One reason for this is the very early adoption phase of smart city solutions. Another reason for this can be found in the knowledge gap between the current city engineers and the smart city community [13].

Using engineering approaches from other domains, such as production and smart grids, is appealing. The current engineering process standards, such as ISO 81346, ISO 15288 [14,15], and others, do not consider smart city properties as indicated above. Thus, this paper can make a contribution to this open field of knowledge.

The SoS literature holds some interesting work related to SoS SOA engineering. Most of the engineering-related publications address SoS SOA engineering at higher levels in enterprises, prominent examples are [16,17]. Very few SoS SOA engineering papers related to production have been found, e.g., [18].

For each of the related requirement areas, some primary references are indicated below:

- Architecture for capturing, e.g., design, engineering, operation, and evolution. RAMI4.0 and IIRA [19,20] are such architectures for Industry 4.0 and, e.g., IEEE 1547 and IEEE2030 [21], which address the smart energy grid domain. There are a number of proposals for smart city high level architectures, see, e.g., [22,23], which seem to be quite simple compared to existing industrial and smart grid architectures.
- Operation of smart city solutions. Many experiments with single functionalities have been pursued. Experiments with multiple functionalities seem to be limited.
- Maintenance of smart city solutions. In this regard, substantial work on maintenance based on SOA and microservices addressing the production domain exists, see, e.g., [24–26]. The maintenance of microservice-based systems is reported by Tizzel et al. [27].
- Training related to smart city engineering, operations, maintenance, and evolution. This is an open field, primarily because of the immaturity of the market for smart city solutions.
- Security of smart city solutions. A comprehensive review can be found in [28].

Since the literature on engineering of smart city functionalities is limited, this paper is intended to provide a valuable contribution to the field.

3. SoS Engineering

Here, the engineering of IoTs and their interaction to form a System of Systems, SoS, in a complex smart city context is addressed. For this purpose, a smart streetlight use case is considered. Thus, the engineering of a complex SoS based on smart streetlight IoTs is elaborated in this paper.

The SoS engineering approach applied is model-based systems engineering (MBSE) [29]. To enable a structured approach to SoS engineering, a feasible engineering process is needed. The one adopted herein comes from Urgese et al. [10], and is depicted in Figure 1. This engineering process can be described using an SOA perspective. The use case architecture is based on SOA/microservices; thus, it follows the major architecture trends in the area. Further discussion of a smart city engineering process makes use of an SOA framework and implementation platform. The engineering process is based on some very recent developments in SOA modelling with SysML [30]. Thus, further description and discussion of the engineering process makes use of Eclipse Arrowhead framework and implementation platform [8,12]. This is facilitated by the available SysML profile and library for Eclipse Arrowhead (http://www.github.com/eclispe-arrowhead/SysML-profile-library (accessed on 29 April 2021)).

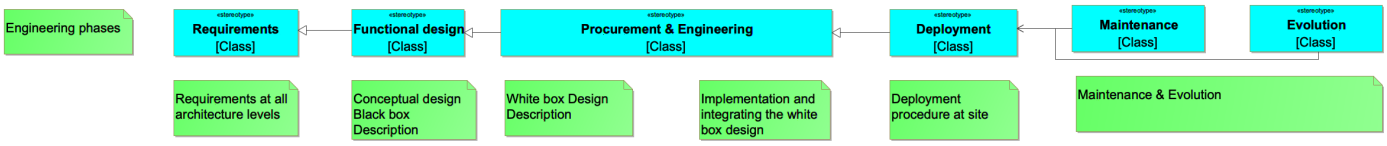

Figure 1. The engineering process applied in this paper is based on ISO 81346 as extended and refined by Urgese et al. [10].

3.1. Smart City Application-Requirements and Evolution

The proposed engineering approach is discussed for a smart streetlight application. Here, each streetlight has a sensor that can detect the presence of a non-stationary object within the illumination area of the streetlight. The "smartness" is that streetlights should only be ON if there is a non-stationary object within its illumination area. It is assumed that there is a small overlap between the illumination areas of each streetlight. The handover of illumination should be made between streetlights when objects are moving from one illumination area to another.

To include the evolution aspect of the application, three generations of the streetlights' sensors will be considered:

Gen.1 IR sensor;
Gen.2 Camera;
Gen.3 3D camera.

Given these streetlight generations, the following scenario regarding design, installation, updates, extension, and segment integration is considered from an engineering perspective:

1. Smart streetlights generation 1 deployed in multiple city sections. The primary objective is that streetlights should be ON when a person or moving object (e.g., bicycle, car, wheelchair) enters the streetlight illumination area.
2. Smart streetlights generation 2 deployed in multiple city sections. In this case, both presence and object location within the illumination area can be detected. The primary objective is still that streetlights should be ON when a person or moving object (e.g., bicycle, car, wheelchair) enters the streetlight illumination area. The secondary objective is that the next street along the position trajectory of the object can be lit up before the object enters that streetlight illumination area.
3. Smart streetlights generation 3 deployed in multiple city sections. In this case, in addition to presence and object location, velocity (speed and direction) within the illumination area can be detected. The primary objective is still that streetlights should be ON when a person or moving object (e.g., bicycle, car, wheelchair) enters the streetlight illumination area. The secondary objective is that the next street along the position trajectory of the object can be lit up before the object enters that streetlight illumination area. The third objective is improved and smoother illumination handover between light poles, resulting in improved illumination experience and reduced energy consumption.
4. Integration between city sections using all smart streetlight generations. Here, we create a handover between city sections, allowing for seamless movement between city sections having different technology generations.
5. Streetlight sections with cameras also provide the emergent capability of tracing the movements of identifiable individuals, which might be of interest as forensic information. This adds engineering requirements on, e.g., data security, storage, and lifetime.

The emerging installation plan for the smart streetlight areas is schematically depicted in Figure 2.

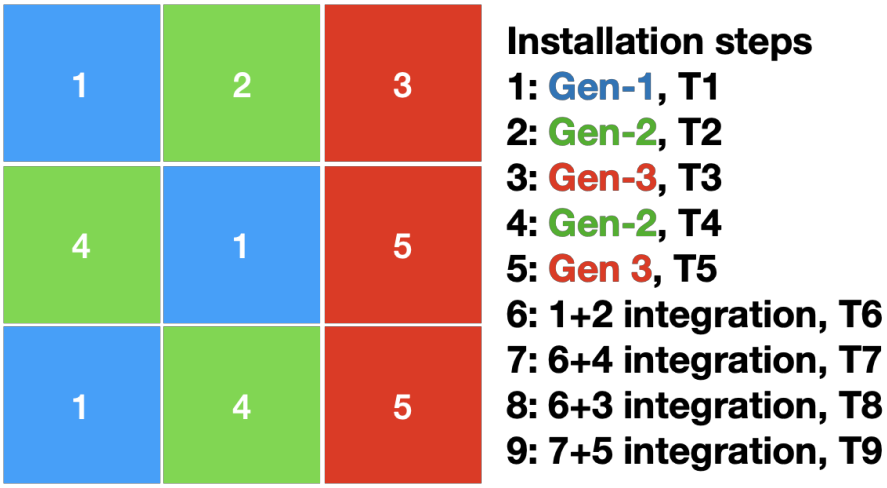

Figure 2. Smart streetlight installation plan for different sections in the city. The smart streetlights generation, section installation time, and section-to-section integration time is indicated for each city section.

3.2. Engineering Approach

The engineering approach proposed herein is based on a two-dimensional perspective. The two dimensions considered are as follows:

- Architecture;
- Engineering process.

The engineering process dimension was briefly introduced above and depicted in Figure 1.

The architecture is based on the Eclipse Arrowhead. Fundamental parts of the architecture are depicted in Figure 3.

Core parts of the architecture are as follows:

- Software system: a software-based microsystem capable of producing and/or consuming microservices and executing its own functionality.
- Local cloud: a set of microsystems in a private network capable of jointly executing a set of functionalities. The local cloud always includes mandatory core microsystems to establish necessary SOA infrastructure. This is a local-scale SoS.
- System of local clouds: a set of local clouds capable of jointly executing a set of functionalities. This is a large-scale SoS enabling the architecture and its engineering to address the expected very large scale of smart cities.
- Device: a hardware hosting one or several software systems.
- Network: a network enabling private clouds with DMZ (De-Militarised Zone) and open internet with communication between the involved software systems and local clouds.

The engineering process, on one hand, and the architecture based on Eclipse Arrowhead, on the other hand, have been used to form a two-dimensional matrix approach to the engineering of complex SoS SOA-based smart city solutions; see Figure 4. The matrix has been captured in a SysML profile [31]. In addition, most of the Eclipse Arrowhead core SOA systems have been modelled and made available as a library, along with a number of templates for often-used constructs. The SysML profile, library, and templates are available at www.github.com/eclipse-arrowhead/profile-library-SysML (accessed on 29 April 2021).

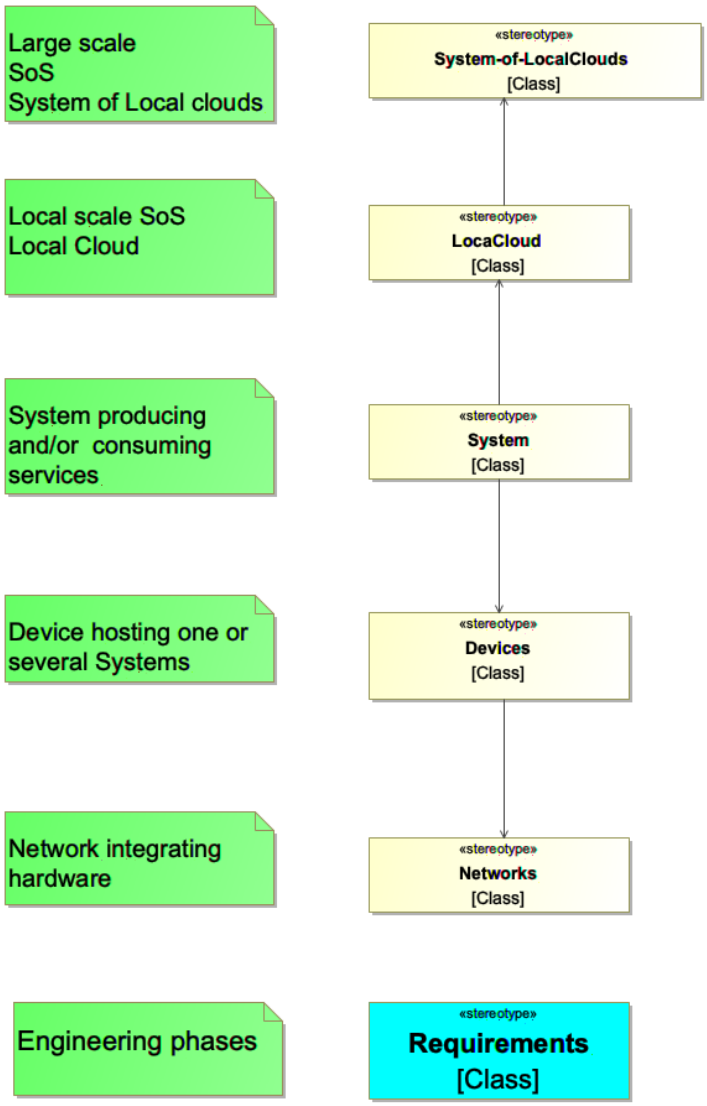

Figure 3. The SoS architecture applied in this paper, Eclipse Arrowhead. It addresses five levels from communication network and devices over microsystems and microservices enabling first level of scalability into individual local clouds and finally to large scalability considering system of local clouds.

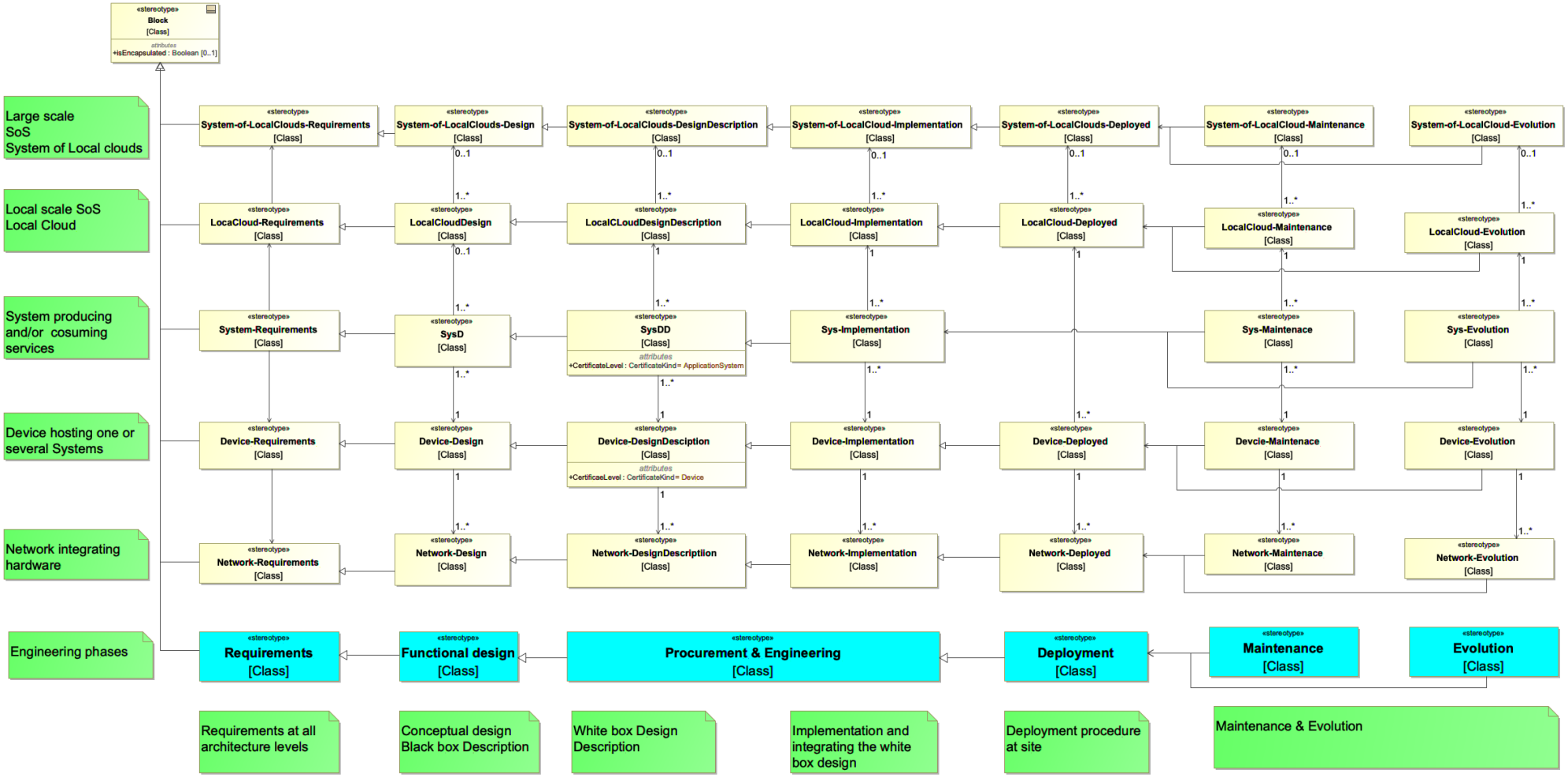

Figure 4. The two engineering approach dimensions. From bottom row up—the architecture dimension starting at the network and device levels up to large-scale SoS level. Thus, enabling the capturing of the very large scale expected in smart cities. The engineering dimension starts with requirements, high-level functional objective, and design, all the way to deployment, maintenance, and evolution. Architectural block relationships and associated constraints are modeled with SysML directed associations.

The Eclipse Arrowhead SysML library provides all currently released core systems of Eclipse Arrowhead plus a number of the release candidate systems (expected release is Q3 2021). In this most fundamentals of SOA/micro-service architectures are supported including;

- Security:Authentication, authorisation, and accounting are supported down to individual service exchanges. Payload protection is provided based on chosen protocol, but for most modern protocols, TLS is used. Secure on-boarding for both hardware and software is supported. Monitoring of security standard compliance and some security issues is also supported by the Eclispe Arrowhead framework and the modelling thereof.
- Interoperability: Autonomous protocol translation, support for data model translation based on ontologies. Adaptors for multiple legacy protocols like, e.g., OPC-UA, Z-wave, Modbus-TCP are provided both as code and models by the Eclipse Arrowhead framework.

An extensive discussion of the SysML modelling of SOA/micro-service and SoS where Eclipse Arrowhead has been used as the validation implementation framework will be provided in a forthcoming paper [30].

Next, each of these dimensions is described in more detail.

3.2.1. Architecture Dimension

Here, we take an SoS architecture perspective and assume a solution approach based on service-oriented architecture and microsystem and microservices. In this context, a microservice is produced by a microsystem, which performs its functionality independently and can be stateful or stateless. The perspective is based on the Eclipse Arrowhead framework and makes use of network, devices, and microsystem with microservices, which are aggregated to independent local clouds which can be further integrated to a system of local clouds. The architecture is divided into these five levels, shown in Table 1.

Table 1. The architectural view of the engineering approach. Here, the use case level and scale are indicated in green boxes with the corresponding architectural level from the Eclipse Arrowhead as stereotyped in SysML in yellow boxes.

Architecture Level	SOA/Eclipse Arrowhead Concepts	Smart Streetlight Use Case Concepts
Network functionality connecting hardware and its functional systems and services	Private local cloud networks with functionality enabling secure service exchanges to other local clouds	Local cloud routers, switches, cabling, wireless access points and access point to the open Internet
Hardware hosting one or more systems and associated services	Hosting one or more microsystems and associated services plus necessary SOA administrative and support systems	The sensor, controller, and light switch, and the necessary SOA administrative and support systems
Individual system (IoT system) and its produced and consumed services	Individual microsystems and their produced and consumed microservices	Individual streetlight poles and their sensor, controller, and the light source switch
Local-scale SoS composed of a number of microsystems	Local clouds' private networks composed of a number of microsystems, each private network responsible for a set of functionalities that primarily can be performed in isolation	City section, e.g., a number of city blocks or streets and associated streetlights
Large-scale SoS interaction between the local-scale SoS networks responsible for a set of functionalities	System of local clouds: aggregated interaction between a number of local clouds and their private networks. In this way scalability to very large number and heterogeneous "smart devices and functionalities" is provided [32].	System of city sections, e.g., sets of streetlight city sections, thus enabling the capturing of very large number of "smart devices" and associated functionalities.

3.2.2. The Engineering Process

The engineering process is based on ISO 81346 and extended to the Eclipse Arrowhead SoS engineering process [10], depicted above in Figure 1. The process is detailed in Table 2.

Table 2. The engineering process view of the engineering approach.

Engineering Phase	SOA/Eclipse Arrowhead Phase	Smart Streetlight Use Case Phase
Requirements	Formalised requirements for all architecture levels	Use case requirements for all architecture levels. As a part of the requirements, the SysML use case diagram model is provided in Figure 5
Functional design	Black box functional designs and models for each architecture level. Here, the system level details of one involved microsystem and its microservice are provided in Figure 6	Use case design and models for each architecture level
Functional engineering and procurement	White box design, models, and engineering for each architecture level, procurement of specified HW, SW, and services. Detailed modelling of functional and security policies, tests, and software instalment procedure. In Figure 7, the white box design of a local cloud is provided.	White box design, models, and engineering for each architecture level of the streetlight solution, procurement of streetlight poles and their sensor, controller, light switch and light source, network, etc. Detailed modelling functional and security policies, test, and deployment procedure
Implementation	Implementation, hardware, and code of the respective microsystems and deployment of code to the dedicated hardware	Implementation, hardware, and code of the smart streetlight microsystems and deployment of code to the sensor, controller, and light switch
Deployment	Physical deployment of hardware, software, functionality and security policies, network, etc. Final tests and operational commissioning according to procedures	Physical deployment of smart streetlight hardware, orchestration and security policies, network, etc. Final tests and operational commissioning according to procedures
Maintenance	Maintenance procedures and execution at all levels of the architecture and its implementation	Maintenance procedures and execution at all levels of the smart streetlight architecture and its implementation
Evolution	Functional and technology evolution and its execution at all levels of the architecture and its implementation	Functional and technology evolution and its execution at all levels of the smart streetlight architecture and its implementation

3.3. Engineering Details

This section details some of the engineering aspects and demonstrates a reasonably possible engineering flow through the architecture/engineering matrix, as shown in Figure 4.

Functional Design-Black Box

Taking the SysML modelling approach, the requirements are first transferred to a use case diagram. In the case of the smart streetlights, an example is shown in Figure 5.

Functionality modelling needs to start with individual systems and their produced and consumed services. This is exemplified herein by a model of the streetlight controller; see Figure 6.

When all black box designs of all involved microsystems and their produced/consumed microservices have been completed, we can move to the next level, namely, local SoS, which in Arrowhead architecture will be treated as a local cloud [32].

Here, the microsystem interactions-exchanges of microservices is modelled. Integrating the involved IoTs into an SoS based on SOA, we also need a model of the necessary SOA infrastructure to support fundamental SOA properties like lookup, loose coupling, and late binding, plus the important IoT security. An example is shown in Figure 7. This model now constitutes a functional Arrowhead local cloud having support for fundamental SOA properties and security. From our use case perspective, such local cloud can correspond to a city section and its smart street lighting.

The integration of functionality in one city section, namely, the local SoS architecture level, is done using orchestration of service exchanges and creation of associated security policies. Orchestration rules and security policy details can be automatically extracted and deployed from an engineering model, such as the SysML model used herein.

It is clear that multiple local clouds can be designed. Interaction between streetlight sections can be established using the Eclipse Arrowhead Gatekeeper and Gateways core systems [33].

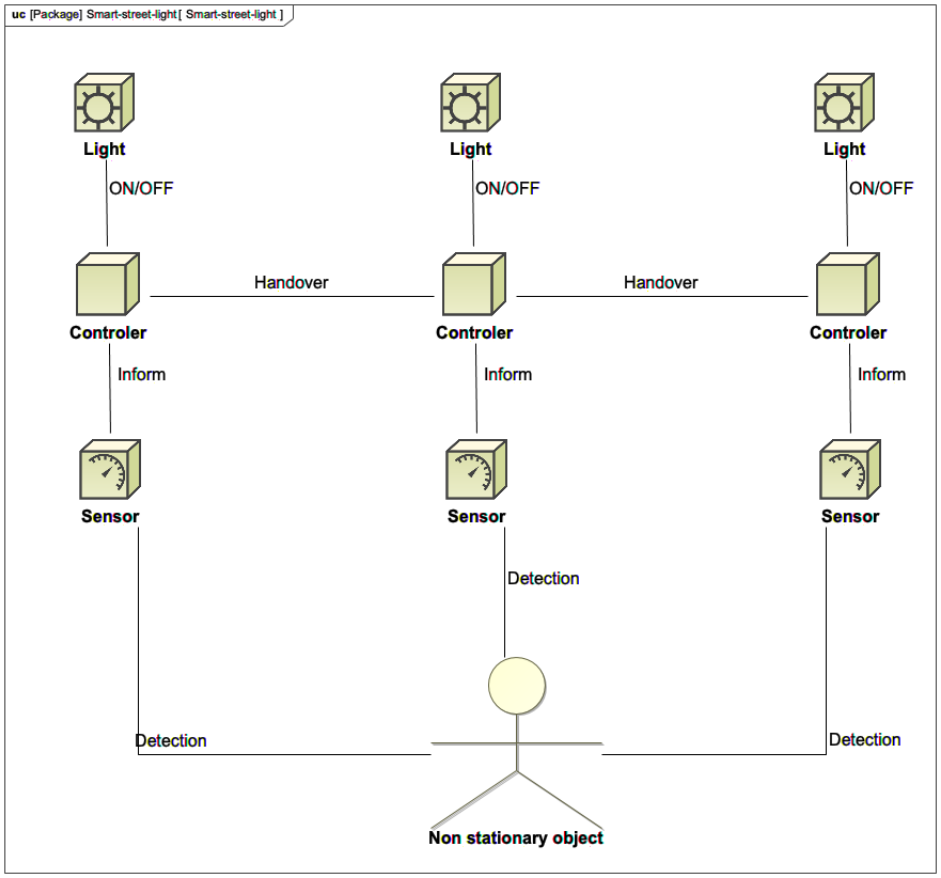

Figure 5. A SysML use case diagram for the smart streetlight use case.

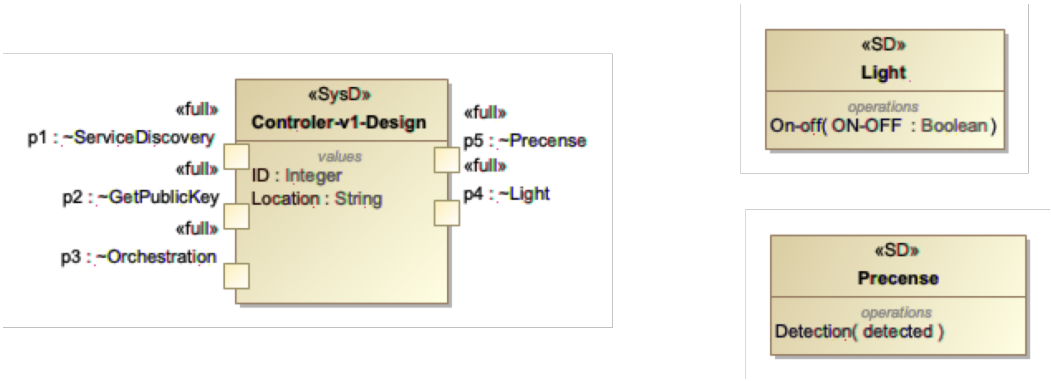

Figure 6. The SysML black box model of the streetlight controller and its consumed application services.

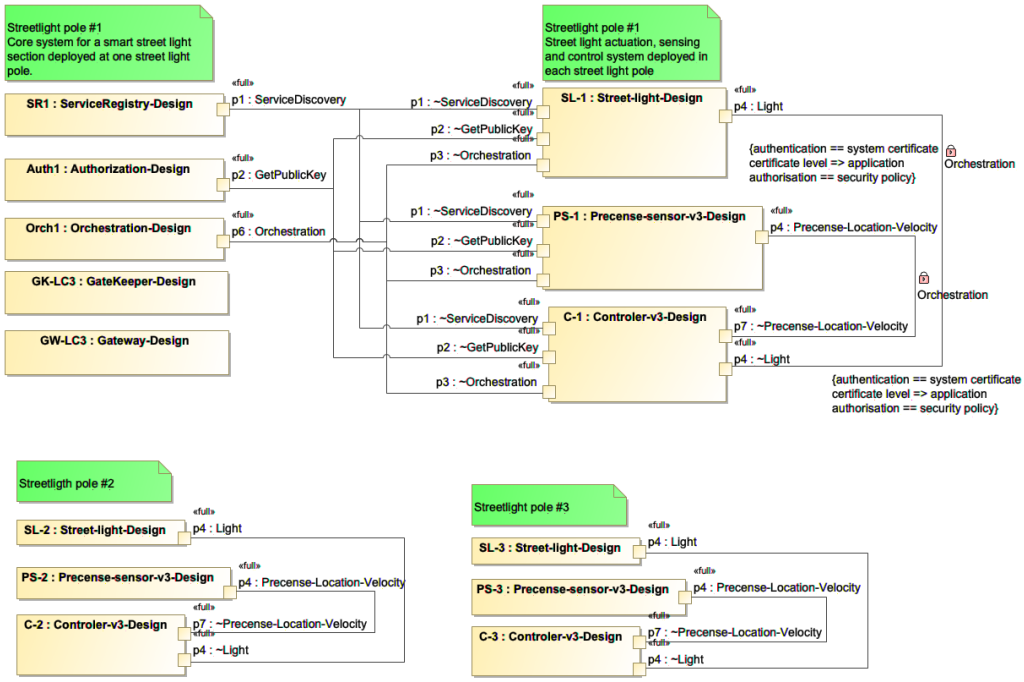

Figure 7. A SysML internal block diagram for a small local cloud of smart streetlight poles, including necessary SOA infrastructure support systems, orchestration principles, and service interaction security policies. For simplicity, only 3 streetlights have been modelled. Each streetlight has the capability of sensing the presence of objects for which the light should be turned on. Only one streetlight holds the Eclipse Arrowhead core systems ServiceRegistry, Authorization, and Orchestration, which supports the fundamental SOA principle. With the help of these core systems, more streetlights can be added, and defective ones can be replaced with minimal system integration effort and cost. In addition, interaction with other streetlight sections can be established using the Eclipse Arrowhead Gatekeeper and Gateways core systems [33].

The next step is to provide the white box design of our involved microsystem, its services, and its interaction within the local cloud. This includes definition of used service protocols, e.g., HTTP, CoAP, MQTT. Security policies for each designed service exchange are also defined at this stage. These definitions will enable partly automated code generation for the designed systems, thus supporting the coding of non-existing application systems. The Eclipse Arrowhead core systems used can simply be pulled from GitHub as docker containers.

Next, the created and identified micro-systems should be installed into some hardware. The hardware should in turn be connected to a network. Thus, we need a model of the hardware devices and the network design. Examples of black box, white box, and specific purpose hardware implementations are provided for hardware devices in Figure 8 and for networks in Figure 9. These models are required to provide necessary information for the deployment of the solution in the real world.

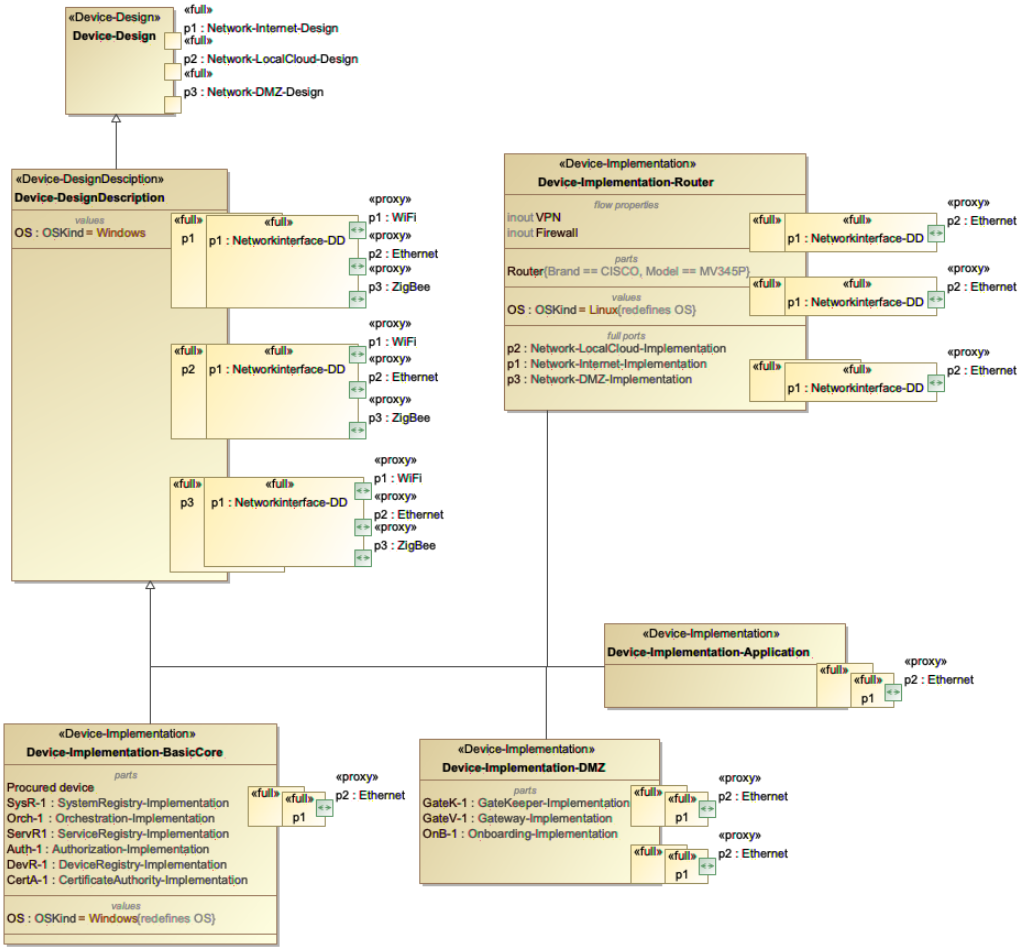

Figure 8. Black box, white box, and implementation models of the hardware devices with some specific implementation examples indicating different device roles and the associated code deployment.

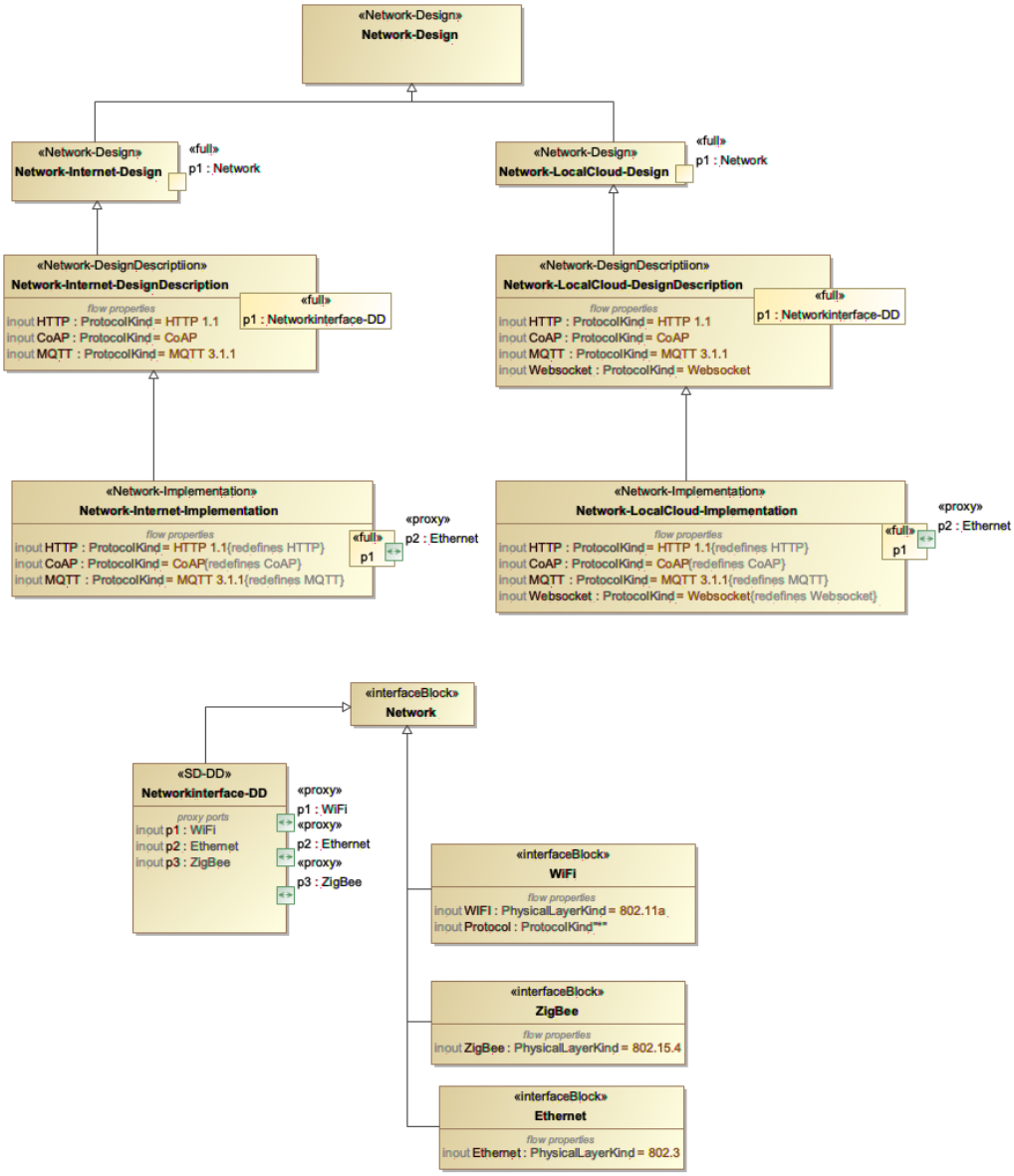

Figure 9. Black box, white box, and implementation models of the open internet structure and the private network structure.

Next, the devices and network are integrated into a local cloud model that describes in which light pole individual hardware and its microsystems are installed. An example is depicted in Figure 10.

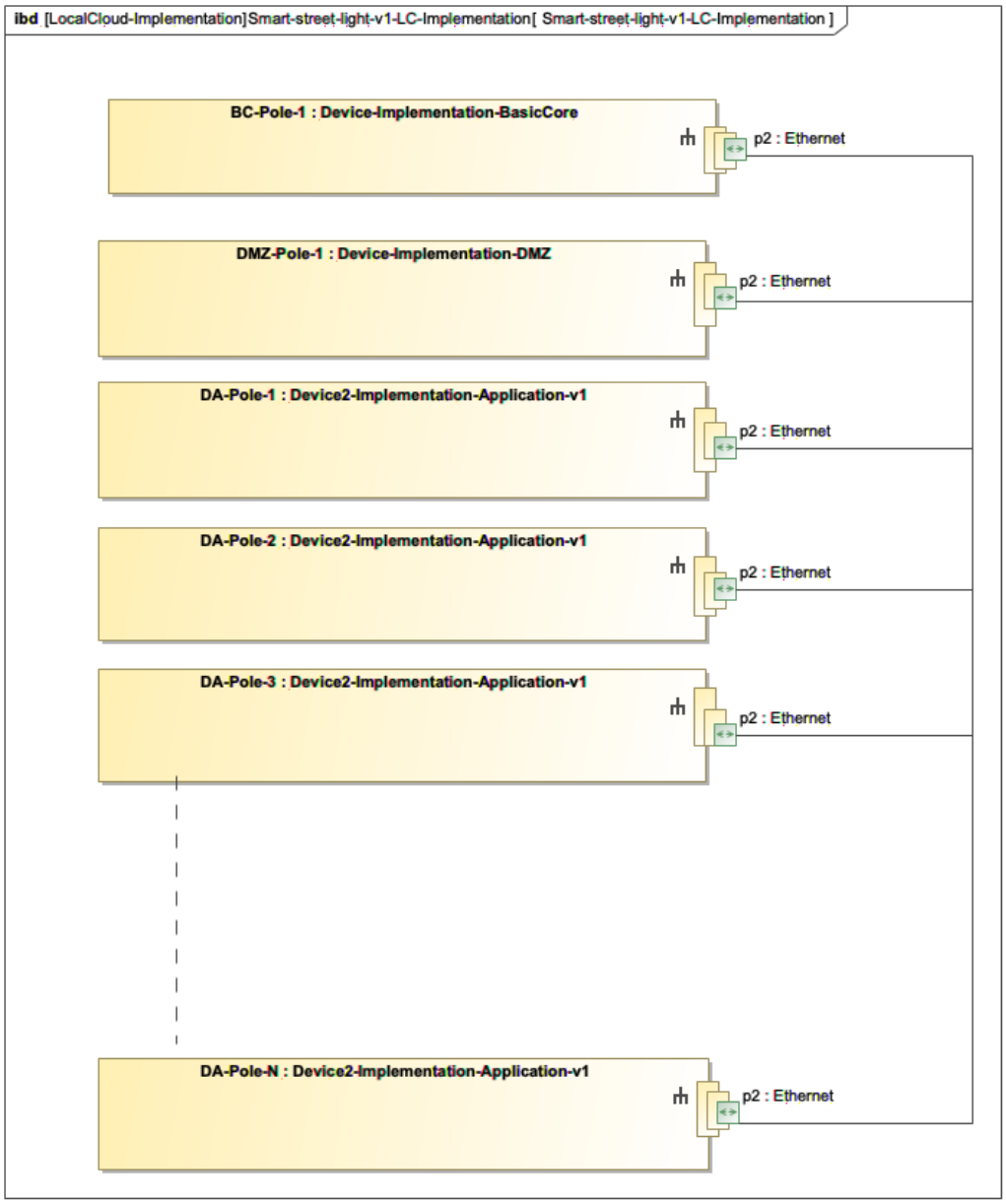

Figure 10. An implementation of a small smart streetlight city section. The designed streetlight microsystems are installed in selected hardware, the SOA infrastructure microsystems are installed on their selected hardware and the local cloud network router is included.

The final step is deployment. In this step, a number of metadata defining where and how each light pole is connected to power supply, network, etc., are provided.

The proposed engineering flow through the architecture–engineering process matrix is depicted in Figure 11.

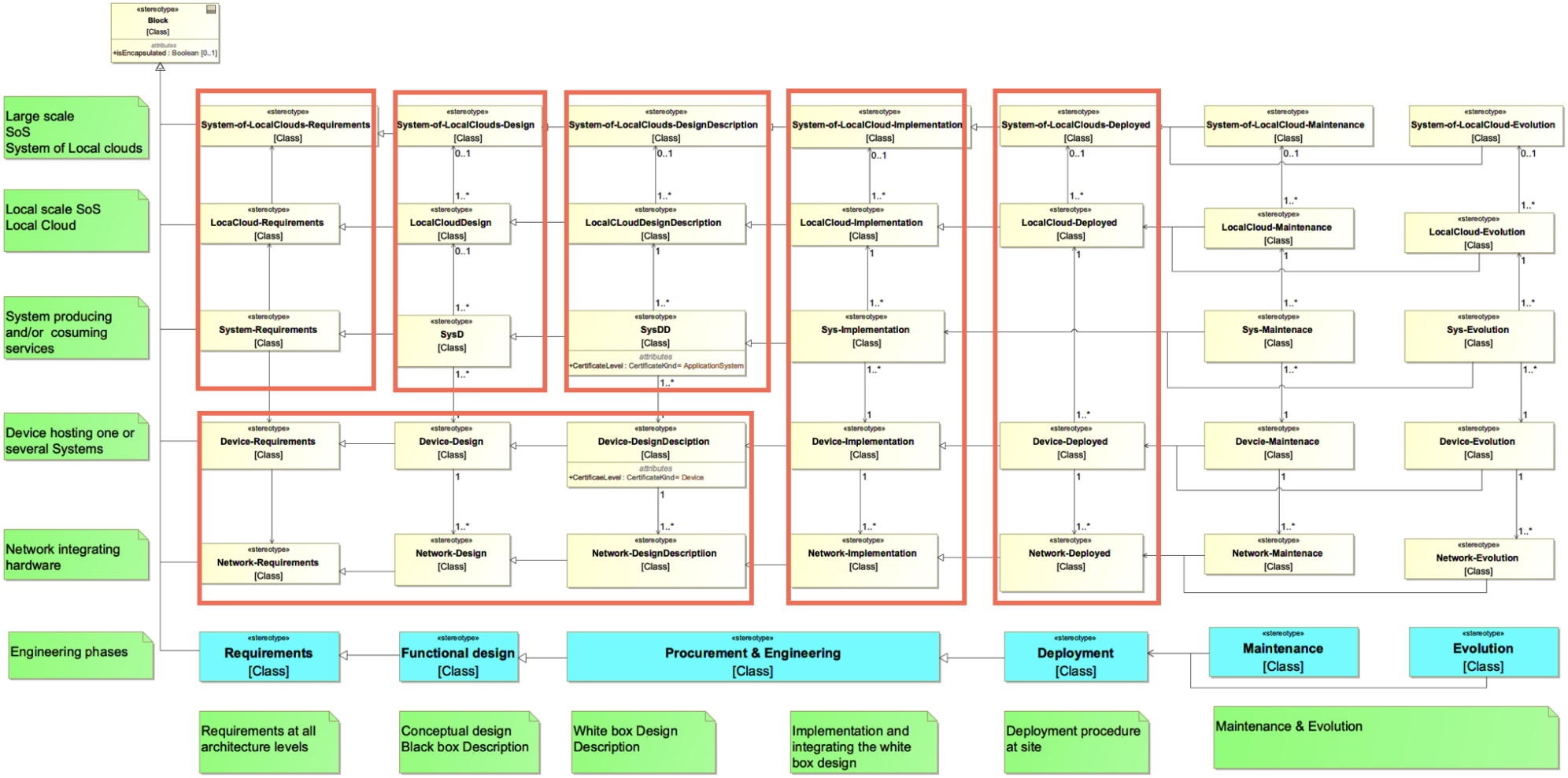

Figure 11. The proposed engineering flow through the architecture–engineering process matrix.

4. Discussion

The proposed engineering approach addresses the knowledge gap identified in the related work Section 2. The very limited smart city engineering literature addresses engineering at design time. This paper provides an approach that supports not only design time engineering, but also in-time engineering, maintenance time engineering, and evolution time engineering. The approach can, to a very large extent, be implemented in SysML engineering tools such as MagicDraw [34] or Eclipse Papyrus [35]. We can identify a number of engineering actions in this approach that can be automated. Such automation has the potential to significantly reduce engineering efforts and cost of smart city features.

Above, the engineering approach for the first smart streetlight deployment in a city sector based on Eclipse Arrowhead architecture was described. To indicate engineering efficiency and use case evolution, the use case comprises three generations of deployment in different city sectors, as shown in Figure 2. The new generations of technology only require the additional modelling of generations 2 and 3 of the sensors and controllers. This is easily done by cloning the 1st generation models and adding the new functionality. Then, we can clone the first city sector local cloud and exchange the sensors and controllers to the new generations. In this way, it is possible to duplicate the engineering models for any number of city sections where smart streetlights are desired.

However, what about interactions between adjacent city sectors? Can handovers between city sectors (see Figure 2) be made using the technology approach? Making use of the Eclipse Arrowhead features, integration of functionality between local clouds and, thus, city sections is clearly possible and maintains security level. The required functionality is provided by the GateKeeper and Gateway systems [36], as indicated already at the functional design level, cf. Figure 7. In this way, service information can be exchanged between different city sections, thus enabling handovers from one city section to an adjacent city section, provided that the necessary functionality is supported by the controllers. Above is another example of SoS evolution and emergent behaviour, which is possible to address and engineer with the proposed approach of this paper. Notably, this type of integration across different technology frameworks is still an interoperable and composability challenge.

Integration between clouds with different technology generations can be managed with minimal engineering efforts, if proper requirements on the services data models are stated. This indicates that technology evolution can be handled positively, given the integration of the engineering process and the SOA-based architecture. These details will be the subject of upcoming papers.

The street light use case and its plan for different sections and technology generations provides a basis for showing the benefits and capability of the engineering process and architecture matrix. The use case construct enables us to show that the following parts of the engineering process can be captured:

- Requirements;
- Functional design;
- Engineering and procurement;
- Deployment;
- Maintenance;
- Evolution.

It further contains all the architecture levels of the architecture. Network and device level to a lesser extent but micro-system and associated micro-service, local cloud, and system of local clouds to a large extent.

For the smart streetlight scenario with its sensing capability, it is clear that other usage scenarios can be added to the light controllers and the data they can collect. Such extended functionality includes, e.g., surveillance information (of potential interest in forensic investigations) and capturing of city usage trends (of interest for city planning).

It is clear that integration of various smart city functionalities using interoperable and composable technology will provide numerous opportunities and risks. Thus, hav-

ing a technology engineering approach that produces documentation of the design, its deployment, maintenance, and evolution is of value when considering both opportunities and risks.

Coming back to the distinguishing properties of smart cities these are supported by the proposed architecture and engineering procedure as follows:

1. Very large number of devices and users >10^5;
   Is enabled by the chosen architecture through the use of the local cloud concept where functional and security scalability is provided to system of local clouds.
2. Very large number of independent functionalities with emerging behaviours and interactions;
   Is supported as indicated with the three generations of sensors in the smart street light use case. Since the architecture and the engineering process is independent from the choice of device and functionality technology, the approach can handle technology with a very large number of independent functionalities. Emerging behaviour and interactions are also supported as discussed above.
3. Extremely heterogeneous technologies;
   (see 2) above.
4. Extremely distributed environments >10 km^2;
   (see 1) above. The only part not discussed is the choice of networking infrastructure enabling extremely distributing environments. Since the 4G and 5G telecom network provides such capabilities, this can be addressed by the engineering approach by adding such network interfaces.

In this way, the utilised SoS architecture, Eclipse Arrowhead, addresses the smart city distinguishing properties, while also providing support for smart city detailed requirements areas like:

- Maintenance of smart city solutions;
- Evolution of smart city solutions;
- Smart city security,

as discussed above.

5. Conclusions

An approach for the engineering of complex smart city solutions has been proposed herein. The engineering spans from design-time over run-time to maintenance and evolution time, thus extending most existing engineering processes to cover almost the whole life cycle of solution. This approach provides a structured integration between a basic SoS architecture applied to smart city use cases and a comprehensive SoS engineering approach based on SOA. The engineering approach has been demonstrated using a smart streetlight solution and its extension and evolution over time. The solution has been modelled in SysML, where it has been shown that functional properties can be addressed. It has also been shown that security policies can be captured in the model and extracted for subsequent deployment and operational use.

The evolution of SoSs and their emerging behaviours have been discussed. It has been noted that the model-based approach provides comprehensive documentation of the SoS solutions. Such documentation will become a valuable asset in the analysis of opportunities and risks connected to integration of various smart city functionalities using interoperable and composable technology.

Funding: Research leading to these results has received funding from the EU ECSEL Joint Undertaking under grant agreement no:826452 (project Arrowhead Tools) and from the national funding authority Vinnova on behalf of Sweden.

Informed Consent Statement: Not applicable.

Acknowledgments: The SysML parts of this paper are based on a fruitful cooperation with Geza Kulcsar, InQueryLabs, Hungary, and Øystein Haugen, University college, Østfold, Norway.

Conflicts of Interest: The author declares no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Maier, M.W. Architecting principles for systems-of-systems. *Syst. Eng. J. Int. Counc. Syst. Eng.* **1998**, *1*, 267–284. [CrossRef]
2. Boardman, J.; Sauser, B. System of Systems-the meaning of of. In Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, USA, 24–26 April 2006; p. 6. [CrossRef]
3. Hidayat, T.; Kurniawan, N.B. Smart city service system engineering based on microservices architecture: Case study: Government of tangerang city. In Proceedings of the 2017 International Conference on ICT For Smart Society (ICISS), Tangerang, Indonesia, 18–19 September 2017; pp. 1–7. [CrossRef]
4. Paniagua, C.; Delsing, J. Interoperability Mismatch Challenges in Heterogeneous SOA-based Systems. In Proceedings of the IEEE International Conference on Industrial Technology (ICIT), Melbourne, VIC, Australia, 13–15 February 2019.
5. Erl, T. *Service-Oriented Architecture: Concepts, Technology & Design*; Prentice Hall: Hoboken, NJ, USA, 2005.
6. Colombo, A.W.; Bangemann, T.; Karnouskos, S.; Delsing, J.; Stluka, P.; Harrison, R.; Jammes, F.; Lastra, J.L. Industrial cloud-based cyber-physical systems. *IMC-AESOP Approach* **2014**, *22*, 4–5.
7. Karnouskos, S.; Colombo, A.W. Architecting the Next Generation of Service-based SCADA/DCS System of Systems. In Proceedings of the IECON 2011, Melbourne, VIC, Australia, 7–10 November 2011; p. 6.
8. Delsing, J. (Ed.) *IoT Automation-Arrowhead Framework*; CRC Press: Boca Raton, FL, USA, 2017; ISBN 9781498756754.
9. Delsing, J. Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions. *IEEE Ind. Electron. Mag.* **2017**, *11*, 8–21. [CrossRef]
10. Urgese, G.; Azzoni, P.; von Deventer, J.; Delsing, J.; Macii, E. An Engineering Process model for managing a digitalised life-cycle of products in the Industry 4.0. In Proceedings of the NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 20–24 April 2020; pp. 1–6. [CrossRef]
11. Bell, M. *Service-Oriented Modeling: Service Analysis, Design, and Architecture*; Wiley & Sons: Hoboken, NJ, USA, 2008; ISBN 978-0-470-14111-3. [CrossRef]
12. Varga, P.; Delsing, J. *Eclipse Arrowhead*; Technical Report; Eclipse Foundation: Ottawa, ON, Canada, 2021.
13. Cosgrave, E. The smart city: Challenges for the civil engineering sector. *Proc. Inst. Civ. Eng. Smart Infrastruct. Constr.* **2017**, *170*, 90–98. [CrossRef]
14. *ISO 811346-12:2018*; Standard document; ISO: Geneva, Switzerland, 2018.
15. *ISO/IEC/IEEE 15288:2015*; Standard document; ISO: Geneva, Switzerland, 2015.
16. Stojanovi, Z.; Dahanayake, A. *Service-Oriented Software System Engineering: Challenges and Practices*; Idea Group Inc. (IGI): Calgary, AB, Canada, 2005.
17. Jamshidi, M. System of systems engineering-New challenges for the 21st century. *IEEE Aerosp. Electron. Syst. Mag.* **2008**, *23*, 4–19. [CrossRef]
18. Mennenga, M.; Cerdas, F.; Thiede, S.; Herrmann, C. Exploring the Opportunities of System of Systems Engineering to Complement Sustainable Manufacturing and Life Cycle Engineering. *Procedia CIRP* **2019**, *80*, 637–642. [CrossRef]
19. Adolphs, P.; Epple, U. *Status Report: RAMI4.0*; Technical Report; VDI/VDE-Gesellshact Mess- und Automatisierungstechnik: Düsseldorf, Germany, 2015.
20. Lin, S.W.; Crawford, M.; Mellor, S. The Industrial Internet of Things Volume G1: Reference Architecture. Technical Report, Industrial Internet Consortium. 2016. Available online: http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf (accessed on 29 April 2021).
21. *Standards IEEE 1547, IEEE 2030*; Standard Document; IEEE SCC21; IEEE: Piscataway, NJ, USA, 2021.
22. Wenge, R.; Zhang, X.; Dave, C.; Chao, L.; Hao, S. Smart city architecture: A technology guide for implementation and design challenges. *China Commun.* **2014**, *11*, 56–69. [CrossRef]
23. Gaur, A.; Scotney, B.; Parr, G.; McClean, S. Smart City Architecture and its Applications Based on IoT. *Procedia Comput. Sci.* **2015**, *52*, 1089–1094. [CrossRef]
24. Halme, J.; Jantunen, E.; Hästbacka, D.; Hegedűs, C.; Varga, P.; Björkbom, M.; Mesiä, H.; More, R.; Jaatinen, A.; Barna, L.; et al. Monitoring of Production Processes and the Condition of the Production Equipment through the Internet. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 1295–1300. [CrossRef]
25. Hästbacka, D.; Halme, J.; Larrañaga, M.; More, R.; Mesiä, H.; Björkbom, M.; Barna, L.; Pettinen, H.; Elo, M.; Jaatinen, A.; et al. Dynamic and Flexible Data Acquisition and Data Analytics System Software Architecture. In Proceedings of the 2019 IEEE SENSORS, Montreal, QC, Canada, 27–30 October 2019; pp. 1–4. [CrossRef]
26. Hästbacka, D.; Jantunen, E.; Karaila, M.; Barna, L. Service-based condition monitoring for cloud-enabled maintenance operations. In Proceedings of the IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 5289–5295. [CrossRef]

27. Tizzei, L.P.; Azevedo, L.; Soares, E.; Thiago, R.; Costa, R. On the Maintenance of a Scientific Application based on Microservices: An Experience Report. In Proceedings of the 2020 IEEE International Conference on Web Services (ICWS), Beijing, China, 19–23 October 2020; pp. 102–109. [CrossRef]
28. Gharaibeh, A.; Salahuddin, M.A.; Hussini, S.J.; Khreishah, A.; Khalil, I.; Guizani, M.; Al-Fuqaha, A. Smart Cities: A Survey on Data Management, Security, and Enabling Technologies. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2456–2501. [CrossRef]
29. Micouin, P. *Model Based Systems Engineering: Fundamentals and Methods*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
30. Delsing, J.; Kulcsár, G.; Haugen, O. SysML Modeling of Service-OrientedSystem-of-Systems. *Innov. Syst. Softw. Eng.* **2021**, submitted for publication.
31. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML*; The MK/OMG Press: Burlington, NJ, USA, 2014.
32. Delsing, J.; Eliasson, J.; van Deventer, J.; Derhamy, H.; Varga, P. Enabling IoT automation using local clouds. In Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 12–14 December 2016; pp. 502–507. [CrossRef]
33. Varga, P.; Blomstedt, F.; Ferreira, L.L.; Eliasson, J.; Johansson, M.; Delsing, J.; de Soria, I.M. Making system of systems interoperable–The core components of the arrowhead framework. *J. Netw. Comput. Appl.* **2017**, *81*, 85–95. [CrossRef]
34. MagicDraw, SysML Modelling Tool, Dassault Systems. Available online: https://www.nomagic.com (accessed on 29 April 2021).
35. Eclipse Papyrus-Modeling Environment, Eclispe Foundation. Available online: https://www.eclipse.org/papyrus/ (accessed on 29 April 2021).
36. Hegedus, C.; Kozma, D.; Soos, G.; Varga, P. Enhancements of the Arrowhead Framework to refine inter-cloud service interactions. In Proceedings of the IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 5259–5264. [CrossRef]