Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 1657–1665
DOI: 10.15439/2014F131
ACSIS, Vol. 2

MuSCa: A Multiscale Characterization Framework for Complex Distributed Systems

Sam Rottenberg*, Sébastien Leriche†, Chantal Taconet*, Claire Lecocq* and Thierry Desprats‡
* Institut Mines Télécom/Télécom SudParis, CNRS UMR 5157 SAMOVAR
Email: firstname.lastname@telecom-sudparis.eu
† Toulouse University, ENAC, Email: firstname.lastname@enac.fr
‡ Toulouse University, CNRS UMR 5505 IRIT, Email: firstname.lastname@irit.fr

***Abstract*—Nowadays, complex systems are distributed over several levels of Information and Communications Technology (ICT) infrastructures. They may involve very small devices such as sensors and RFID, but also powerful systems such as Cloud computers and knowledge bases, as well as intermediate devices such as smartphones and personal computers. These systems are sometimes referred to as multiscale systems. The word "multiscale" may qualify various distributed systems according to different viewpoints such as their geographic dispersion, the networks they are deployed on, or their users' organizations. For one entity of the multiscale system, communication technologies, non-functional properties (for persistence or security purpose) or architectures to be favored may vary from one scale to another. Moreover, ad hoc architecture of such complex systems are costly and non-sustainable. In this paper, we propose a scale-awareness framework, called MuSCa. This framework includes a characterization process based on the concepts of viewpoints, dimensions and scales. These concepts constitute the core of a dedicated metamodel. The proposed framework allows multiscale software designers to share a taxonomy for qualifying their own system. At system design time, the result of such a qualification is a model from which the framework produces scale-awareness artifacts. As an illustration of this model-driven approach, we show how multiscale probes are generated to provide multiscale components with an embedded scale-awareness ability.**

***Index Terms*—Multiscale Distributed systems, Model Driven Engineering**

I. Introduction

SEVERAL recent research works [1], [2], [3] consider complex distributed systems that include both very small systems such as objects from the Internet of Things (IoT) paradigm, and powerful systems such as those found in the Cloud. This collaboration enables each system to benefit from the capabilities of the others. Some of these systems also involve intermediate computers such as mobile devices or proximity servers. Those complex systems could also be viewed as *multiscale distributed systems*.

As stated in [4], a "complex system" is any system comprised of a great number of heterogeneous entities, where local interactions among entities create multiple levels of collective structure and organization. Identifying underlying superstructures of complex systems is a challenge. A multiscale analysis of complex systems provides reduced views of those systems with simplified structures, such as presented in [5].

Multiscale distribution is a different concept from large-scale distribution. Large scale has a quantitative meaning, whereas multiscale has an heterogeneity meaning [6], [7]. The system heterogeneity may come from differences of latency or protocols of involved networks, from differences of storage capacity or nature of devices, or from dispersion variations between entities. We propose to study the multiscale nature of a complex system at design time. Approaches that allow developers to work at a high level of abstraction are needed. Model-Driven Engineering (MDE) approaches may help to describe complex systems at different levels of abstraction and from a variety of perspectives [8].

The contribution of this paper is a multiscale characterization framework, called MuSCa (***Mu**lti**S**cale **Ca**racterization framework*). This framework provides a multiscale taxonomy to describe at design time the multiscale nature of complex distributed systems. The first contribution is a multiscale characterization process. It is based on the concepts of viewpoints, dimensions and scales. We follow a model-driven approach to produce a multiscale characterization editor to qualify complex distributed systems. As a second contribution, multiscale probe artifacts are generated for runtime scale-awareness purpose. With those artifacts, system entities become aware of their place in the organization of the system. In the future, multiscale characterization, and multiscale probes may enable software stakeholders to build, deploy, and manage complex distributed systems.

This paper is organized as follows. Section II presents the motivations for a multiscale characterization framework. Then, Section III proposes a generic characterization process for multiscale systems. The MuSCa framework is presented in Section IV. Finally, Section V presents related works, and Section VI concludes the paper.

II. Motivations for a multiscale characterization framework

This Section presents the motivations for a multiscale characterization framework. Section II-A discusses the heterogeneity of complex distributed systems from the ICT infrastructure viewpoint. Afterwards, Section II-B details some motivating examples for multiscale characterization. Finally, based on these motivations, Section II-C outlines our contribution.

978-83-60810-58-3/$25.00

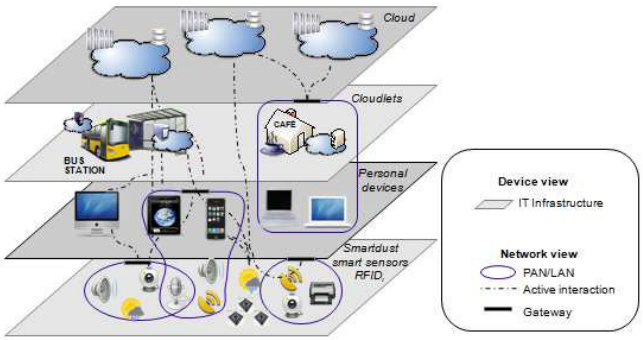

Fig. 1. ICT infrastructure levels of a multiscale distributed system

A. Complex distributed systems and multiscale distributed systems

Fig. 1 depicts our vision of various ICT infrastructure levels that compose complex distributed systems. This figure considers four ICT levels: at the bottom, the smartdust, Radio Frequency Identification (RFID), smart sensors level; at the top, the Cloud computing level; and two intermediate infrastructure levels: firstly personal devices and secondly cloudlets [9] in cafes and bus shelters.

A system, such as the one depicted in Fig. 1 has the topology of a complex system. The interactions between the system entities are decentralized. Many interactions are local but some of them take place between levels. Therefore, the interaction graph between the entities is non trivial, and is difficult to simplify. Moreover this kind of complex systems is composed of a great number of heterogeneous entities, collaborating through different networks, protocols, rules, depending on their respective organizations, geographical distance for instance.

In order to better understand and master the inner complexity of such highly heterogeneous systems, the study of the multiscale nature of a complex system may help to obtain meaningful organizations of a complex system.

B. Motivating scenario

This section presents a motivating scenario that is used throughout the paper in order to illustrate and to evaluate our contribution. This scenario involves a context management system, which is a complex distributed system. This system is deployed through many entities in the city of Toulouse in France. This system aims at enabling a large number of end-users and connected objects to share their context information, such as their location. This scenario is studied through two different aspects: the deployment aspect and the context data filtering aspect.

Considering the deployment aspect, in this scenario it is required that for each local network (LAN) containing at least one context-aware object, one software component dedicated to filter context management data must be installed in the same LAN, on any device having a bandwidth greater than 50 MB/s. Moreover, to get a scalable architecture, another software component dedicated to route context management data must be installed in a hierarchical way, one for each geographical dimension of the city of Toulouse (one for each building, one for each district and one for the city). At last, the routing component for the city of Toulouse must be installed in a cloud and end-user context-aware components must be installed on smartphones.

The second aspect is the context data filtering aspect. In the studied scenario, the user should be able to express constraints about the information he or she wants to receive and about the users that are allowed to receive his or her context information. The two following use cases are studied in this paper. In the first use case, a user, called Sophie, is going to the theater and wants to find a place to park her car. She wants to use the context management system with her smartphone to see the parking places available at foot distance from the theater, where she has just arrived. In the second use case, Sophie wants to share her location, but only with her friends located in her neighborhood.

In this scenario, it is required to express scales of distances (e.g., foot distance, same neighborhood) between system entities—i.e., users and parking places—but also scales of devices (e.g., smartphone, super-computer), network topology, or even geographical administration. These scales constraints drive interactions between entities.

C. From motivations to our contribution

All these use cases motivate the need for a multiscale vision of complex distributed systems. This vision enables a system designer or a user to express constraints concerning different points of view (e.g, geographic dispersion of system entities, network organization, social organization, devices). The solution proposed in this paper is a multiscale vocabulary on which is based a multiscale characterization process for complex distributed systems. This process is then formalized and applied through a MDE approach. The main results of this approach are, firstly, a shared extendable multiscale taxonomy, which can be used to characterize a system or express configuration and behavior constraints, and secondly, generated artifacts, which enable to enforce the constraints at runtime.

III. Multiscale characterization process of complex distributed systems

The MuSCa approach is presented in Section III-A. Then, Section III-B defines a multiscale vocabulary. This vocabulary is illustrated for the geography viewpoint in Section III-C. Other multiscale viewpoints are discussed in Section III-D. Finally, Section III-E presents the full characterization process.

A. MuSCa approach

Fig. 2 depicts the general approach followed by the MuSCa framework. The design process, which is detailed with a SPEM [10] diagram, is composed of two main activities. The first activity is the multiscale characterization to render a multiscale analysis of a complex distributed system. In order

to guide the system designer and to capitalize on previous characterizations, this activity takes the MuSCa taxonomy as an input. This taxonomy contains all the multiscale characterization terms that have already been used in previous characterizations. The result of this activity is a MuSCa model, which is a restriction of the MuSCa taxonomy. If a system designer wants to use new multiscale terms when describing the model, these terms are added to the taxonomy and will be available for the next characterization. In the second activity, multiscale probe artifacts are generated from a MuSCa model. Those probes consolidate data provided by lower level probes, called basic probes. They enable to identify the scales of the system entities at runtime.

B. Multiscale vocabulary

The multiscale characterization process must be based on a precise vocabulary. In the following the concepts of viewpoint, dimension, measure, scale, scale set are defined. These concepts will constitute the structure of the MuSCa metamodel presented in Section IV-B.

The architecture of a system is obtained by studying this system from different *viewpoints*, each viewpoint leads to a view of the system [11].

Each viewpoint is studied through *dimensions*. A multiscale dimension is a measurement of a particular characteristic of a system view for a particular viewpoint.

A dimension is associated to a *measure*, which can be either numeric or semantic.

Using a measure, a dimension can be divided into *scales*. A scale matches, respectively, orders of magnitude for numeric measures, or sets of elements that share common semantic characteristics for semantic measures.

A *scale set* is the set of scales chosen for a given dimension and measure couple.

C. Geography viewpoint

To illustrate the MuSCa vocabulary, Fig. 3 presents possible dimensions and scales for the geography viewpoint. The study of other viewpoints is available for download[1].

For the geography viewpoint, we have chosen to study the multiscale nature of a system through two dimensions, respectively associated with one numeric measure and one semantic measure. The *distance* dimension measures in meter the maximum distance between a set of entities. A set of four scales has been selected for this numeric measure: *local* under $10m$, *footdistance* between $10m$ and 10^3m, *cardistance* between 10^3m and 10^5m, and *planedistance* above 10^5m. For this dimension, the center of each scale is distant from several orders of magnitude from the center of the other scales. The *administrative division* dimension is associated to a semantic measure. It is also applied to a set of entities. It measures their smallest common division. For this analysis, we have selected a set of six scales: *Building*, *District*, *City*, *Region*, *Country* and *World*. One can notice that according to the chosen dimension and scales to study a geography view of the system, the possible organizations and interactions between entities of the system differ.

[1]http://anr-income.fr/uploads/MultiscaleViewpoints.pdf

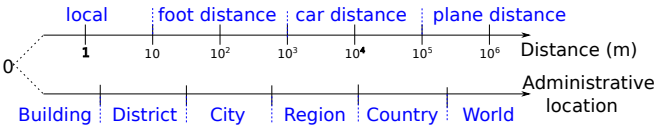

Fig. 3. Dimensions and scales for the geography viewpoint

D. Other multiscale viewpoints

The geography viewpoint is not the only viewpoint to analyze during the multiscale characterization of a distributed system. There are other fundamental viewpoints to consider such as the study of the devices of the system (device viewpoint), the social organization (user viewpoint), the administration organization (administration viewpoint), the network connections between the system entities (network viewpoint). Indeed, they are related to the main issues at stake in the design, implementation and deployment of such complex systems: the need of computing power or storage capacity (device viewpoint), of interaction between distant entities (network viewpoint), and of social organizations (user viewpoint).

However, the above viewpoint list is not exhaustive and other viewpoints, such as data, or time, could also be considered. Depending on the properties to be highlighted for the systems, one may choose to study different viewpoints and dimensions; this is the reason why we propose an open multiscale characterization process.

E. Multiscale characterization process

The multiscale nature of a distributed system should be studied independently from each considered viewpoint. For each viewpoint, a restricted view of the system is considered. Then, for this view, one or several dimensions are chosen. To identify scales for a given dimension, each dimension is associated with a numeric or a semantic measure. Depending on the type of the measure, the resulting scales match, respectively, orders of magnitude for numeric measures, or sets of elements that share common semantic characteristics for semantic measures. The choice of the viewpoints, dimensions/measures and of the scales relevant for a multiscale characterization, is left open depending on the properties of the system one wants to highlight. The objective of these choices is to put to the fore specific characteristics to deal with during the system design, or the system runtime.

The multiscale nature of a system is relative to a multiscale characterization; it is studied independently from each viewpoint. For a given viewpoint, and for a couple dimension/measure, each element of the restricted view of the system is associated with a scale. For a given characterization, and a given viewpoint, a distributed system is qualified as multiscale when, for at least one dimension, the elements of its view are associated with different scales.

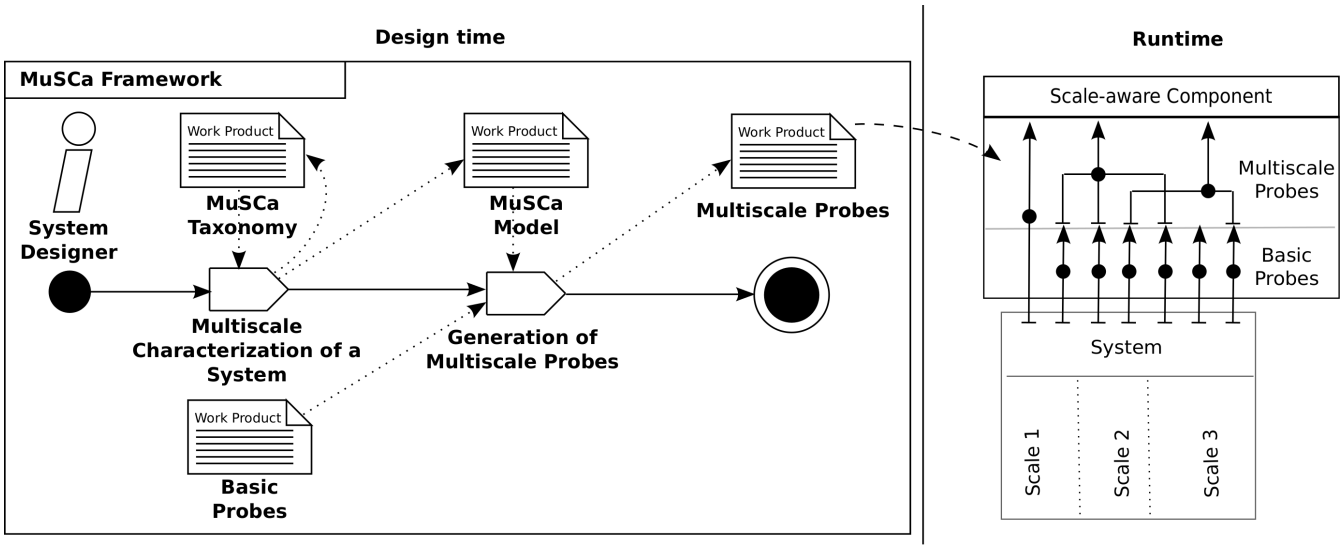

Fig. 2. MuSCa framework approach

IV. MuSCa framework

This section presents the MuSCa framework. Firstly, Section IV-A presents the model-driven approach. Then, Section IV-B formalizes the characterization process with the MuSCa metamodel, and Section IV-C gives an example of a MuSCa model. Thereafter, Section IV-D describes the generated artifacts —i.e., multiscale probes. Finally, Section IV-E presents some MuSCa implementation details and different utilizations of MuSCa are given in Section IV-F.

A. Model-driven approach

In order to formalize the multiscale characterization process, and to use it in the design and deployment of scale-aware distributed systems, we have chosen to follow a model-driven approach (using the four OMG meta-modeling layers [12]). Fig. 4 shows the mapping between the model-driven architecture levels and the MuSCa levels. The MuSCa metamodel (M2 level) is defined with the Ecore meta-metamodel (M3 level). The classes of the MuSCa metamodel represent multiscale concepts. This metamodel is used to define characterization models (M1 level). This characterization may be used for one or several real world systems (M0 level). We also follow the model-driven approach in order to automatically produce artifacts, for instance, producing probe artifacts for scale-awareness.

B. MuSCa metamodel

The MuSCa metamodel is shown in Fig. 5. This metamodel is based on the vocabulary used in the multiscale characterization process.

An instance of MSCharacterization is the result of a characterization process. A characterization considers several Viewpoints —e.g., Geography, User, Device and Network viewpoints (M1 level classes). Each viewpoint determines a restricted view of the system that is studied independently. A

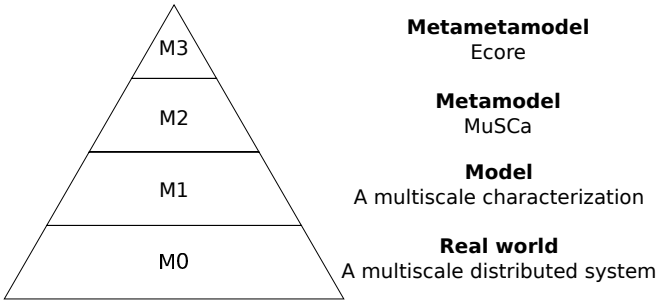

Fig. 4. MuSCa: Model-driven architecture levels

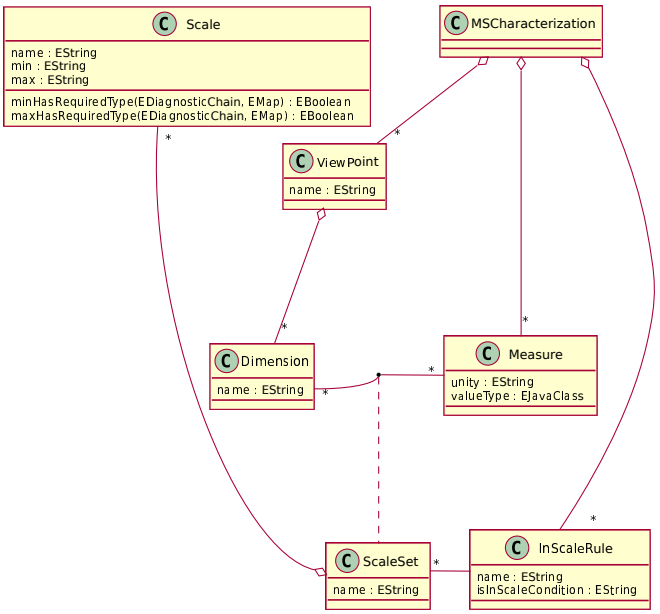

Fig. 5. MuSCa: Multiscale characterization metamodel

view of the system from a given viewpoint is studied through several Dimensions, which are measureable caracteristics of the elements of the view. For example, the Device viewpoint can be analyzed through the StorageCapacity (M1 level) dimension of the system devices. As previously mentioned, a Dimension is measurable, meaning that it can be associated with one or several Measures. For example, at M1 level, the StorageCapacity dimension may be measured with the Bytes measure or the KiloBytes measure. For the association of one dimension with one measure, the designer defines a ScaleSet, which is an ordered set of scales relevant for the system. Each scale set is associated to an InScaleRule, which determines the condition that the measured dimension of an entity, for this scale set, must satisfy so that the entity is associated to a scale. For example, for numeric measures, a Scale is defined by its min and max bounds and a rule can express that an entity is associated to a scale because its measured value is strictly between min and max. Finally, for some viewpoints, the system may present several instances of one scale. For example, if we take the Geography viewpoint, in the AdministrativeLocation dimension, the City scale (M1 level) often has several instances (M0 level) —i.e., the different cities where entities of the system are present.

C. MuSCa model as a multiscale characterization

Fig. 6 illustrates an extract of a MuSCa model that is the result of a characterization process applied to the scenario presented in Section II-B. As mentioned in Section III-A, this model is a restriction of the MuSCa taxonomy. Four viewpoints have been selected: device, network, geography and user viewpoints (the figure only shows the geography viewpoint scales).

We study the geography viewpoint through two dimensions. The first dimension, which we call the "smallest common location" dimension, measures the distribution of the system by studying the smallest common administrative location of a set of scale-aware entities. For this dimension, measured in what we call the "smallest common location measure" (semantic measure), we identify the following scales: building, district, city, region, country, and world. The second dimension, called the "smallest common distance" dimension, studies the distribution of system entities in terms of distances between each other. For this dimension, measured in meters (numeric measure), we identify the following scales: local, foot distance, car distance, and plane distance. Scales are characterized by the min and max bounds in meters. An illustration of these two geography dimensions can be found in Section IV-F, Fig. 7.

D. From MuSCa model to multiscale probes

From a MuSCa model, multiscale probe artifacts are automatically produced. These probes are monitoring programs that are to be deployed on each entity of a multiscale system. One probe is generated for each viewpoint. Each probe exposes at least one method by dimension. This mandatory method returns a scale for a set of entities (e.g., the smallest

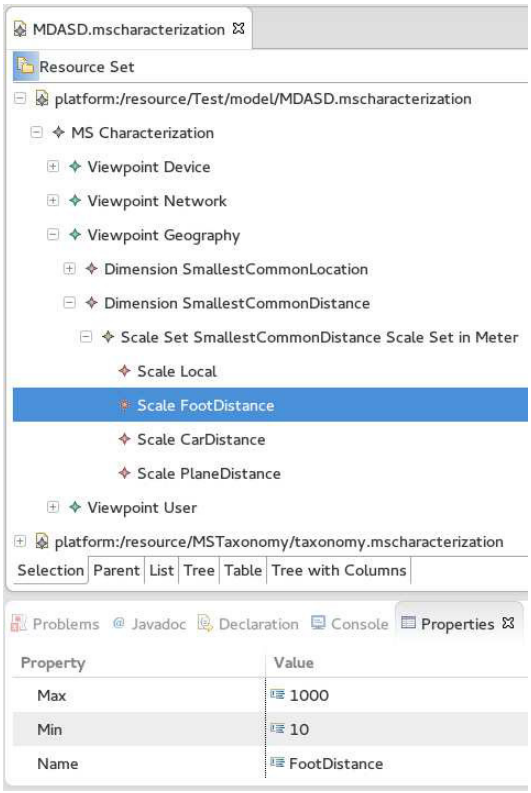

Fig. 6. Example of a MuSCa model

common location measure may return the City scale). For some dimensions, MuSCa also generates one method, which returns the scale instance for a set of entities (e.g., the name of the city). For numeric measures, the methods are automatically generated. The generated methods can be completed to implement a specific logic, in particular to call basic probes, as shown in Fig. 2, or to implement specific semantic measures.

An extract of an automatically generated probe is presented in Listing 1. This probe exposes the getScale method line 5, which uses the getValue method line 3 to call the basic probe. The getScale method returns one of the scales listed between lines 2 and 5 in Listing 2. This last listing contains two generated methods: isInScale method at line 26, which tests if a given measure value is between min and max bounds of a given scale, and getScale method at line 15, which returns the first scale corresponding to a measure value.

```
1  public interface IGeographyMeasurable {

3    public Meter_Value
4      getValue_SCD_In_Meter(List<IGeographyMeasurable> args);
5    public SCD_In_Meter_Scale
6      getScale_SCD_In_Meter(List<IGeographyMeasurable> args);

8    // same methods for SmallestCommonLocation dimension

10 }
```

Listing 1. Extract of generated IGeographyMeasureable interface

```
1  public enum SCD_In_Meter_Scale {
2    LOCAL("0", "10"),
3    FOOTDISTANCE("10", "1000"),
```

```
4     CARDISTANCE("1000", "100000"),
5     PLANEDISTANCE("100000", "Infinity");

7     private final Meter_Value min;
8     private final Meter_Value max;

10    SCD_In_Meter_Scale(String min, String max) {
11      this.min = new Meter_Value(min);
12      this.max = new Meter_Value(max);
13    }

15    public static SCD_In_Meter_Scale
16      getScaleFromMeasureValue(Meter_Value value) {
17        for (SCD_In_Meter_Scale scale :
18          SCD_In_Meter_Scale.values()) {
19            if (scale.isInScale(value)) {
20              return scale;
21            }
22        }
23        return null;
24    }

26    private boolean isInScale(Meter_Value value) {
27      return (value.compareTo(this.min) >= 0)
28        && (value.compareTo(this.max) <= 0);
29    }
30  }
```

Listing 2. Extract of generated SmallestCommonDistance_In_Meter_Scale enumeration

This probe has been generated with the Acceleo[2] code generator. As an example, Listing 3 shows an extract of an Acceleo template. When this template is applied to the MuSCa model presented in Fig. 6, it generates the scales listed between lines 2 and 5 of Listing 2.

```
1 [for (scale : Scale | aScaleSet.scales) separator(',\n')]
2   [scale.name.toUpper()/]("[scale.min/]", "[scale.max/]")
3 [/for];
```

Listing 3. Extract of a MuSCa Acceleo template

E. MuSCa implementation

We have implemented MuSCa with the Eclipse Modeling Framework Project[3] (EMF). The MuSCa metamodel is defined as an instance of the Ecore metametamodel. EMF generates a specialized model editor. We have extended the editor for validation purpose. This editor has been used to define the MuSCa model presented in Fig. 6. Then, the Acceleo code generator is used to produce multiscale probes implemented in Java.

For illustration purpose, we have detailed in this paper the multiscale probe generated for the geography/smallestCommonLocation viewpoint/dimension couple. The probes corresponding to the device/storageCapacity and geography/smallestCommonDistance couples have also been implemented. Moreover, other multiscale probes can be generated through the same process, for any given viewpoint/dimension couple, provided there is an available basic probe for the corresponding dimension.

F. MuSCa in action

So far, this work is used through three aspects. First, MuSCa has been used to study different IoT scenarios in the INCOME[4] project. Then, MuSCa is used to specify and implement multiscale deployment requirements in a multiscale software deployment tool. At last, MuSCa is used to add multiscale requirements between data producers and consumers and generate the multiscale probes used in the implementation. This section presents a synthesis of these three aspects.

1) Multiscale IoT Scenarios analysis: In the INCOME project, in order to characterize the multiscale nature of several IoT scenarios, the MuSCa vocabulary has been used as a reading grid. The scenarios have been analyzed through different viewpoints and dimensions to highlight their revelant scales. This approach enabled the INCOME project members to compare a great variety of scenarios. This work helped to build the MuSCa taxonomy, which was based on the viewpoints, dimensions, measures and scales identified during the study of the scenarios.

2) Multiscale deployment: Deployment of software entities on devices in a multiscale system is another concern of the INCOME project. For this purpose, MuSCa has been used to define MuScADeL [13], a domain-specific language (DSL) dedicated to multiscale and autonomic software deployment. MuScADeL allows deployment designers to abstractly define deployment properties without exact knowledge of the devices and networks the system will be deployed on. MuSCa helps deployment designers to characterize the multiscale nature of a system from several viewpoints such as device, network, administration and geography. An example of the language (deployment of the motivating scenario) is presented in the Listing 4.

```
2   // Definition of probes
3   Probe Network {...}
4   MultiScaleProbe MSNetwork {..}
5   MultiScaleProbe Geography {..}
6   MultiScaleProbe Device {..}
7   // Definition of a criterion
8   BCriterion Crit50MB { Network.bandwidth > 50; }
9   // Definition of deployment requirements
10  Deployment {
11    // deployment of filter components
12    F @ Crit50Mb, Each MSNetwork.Type.LAN;
13    // deployment of hierarchical routing components
14    R @ Each Geography.Location.Building,
15        Geography.Location.City("Toulouse");
16    R @ Each Geography.Location.District,
17        Geography.Location.City("Toulouse");
18    R @ Geography.Location.City("Toulouse"),
19        Device.PowerProcessing.Cloud;
20    // deployment of end-user context-aware components
21    C @ All Device.Type.Smartphone,
22        Geography.Location.City("Toulouse");
23  }
```

Listing 4. Multiscale deployment constraints

As a MuScADeL code is linked to a MuSCa specific model, the MuScADeL editor can check that dimensions and scales conform to the ones defined in the MuSCa model associated with it. In addition, multiscale requirements are verified at runtime by the multiscale probes generated for this MuSCa model.

MuScADeL runs alongside MuSCa, as an Eclipse plugin, allowing the deployment designer to be able within the same

[2]http://www.eclipse.org/acceleo/

[3]http://www.eclipse.org/modeling/emf/

[4]http://anr-income.fr

engineering tool (Eclipse) to define new multiscale viewpoints, dimensions or scales, before using them in the deployment DSL.

3) Multiscale context data filtering: Concerning context data filtering, there are ongoing works in the INCOME project to extend context data routing and filtering with expressions that use the multiscale vocabulary. These scale-aware routing requirements can be added to context data producer and consumer contracts [14]. The aim of these works is to express privacy and quality of context constraint based on multiscale concerns —e.g., to get context data from parking places located at foot distance, or to share context data with user located in the same neighborhood. These constraints are defined from a model of the multiscale characterization of the deployed system, which gives to the contract designers a shared vocabulary restricted to the existing scales in the system.

Once the constraints are defined, they can be enforced thanks to the multiscale probes generated with the model-driven approach. These probes, which are generated from the same MuSCa model as the one used to define the constraints, can characterize a system entity at runtime in order to decide if this entity matches a constraint or not. For example, Fig. 7 illustrates the areas that match the context data filtering constraints presented in Section II-B —i.e., Sophie's neighborhood and foot distance scale instances. This figure was produced with the help of a generated geography multiscale probe, which has been implemented by calling the reverse geocoding API of the OpenStreetMap[5] Nominatim[6] service, and the map has been produced using the JMapViewer[7] API (also part of the OpenStreetMap project).

This example illustrates the interest of the MuSCa metamodel. The multiscale characterization of the system, which contains rules to associate system entities to scales, is expressed in a MuSCa model with a shared vocabulary, in a declarative way. The MDE approach allows to automatically generate the appropriate probes depending on the rules and scales declared in the MuSCa model. As the same MuSCa model is used to express context data routing constraints, the probes can be used to identify the areas that match these routing constraints.

V. Related works

We have noticed the rising presence of complex distributed systems in several recent research studies [15], [1], [2], [3], [16]. Some of these systems are explicitly described by their authors as multiscale [15], [3], [16]. However, we did not find any definition of the *multiscale* vision of distributed systems and when it should be applied. We propose a framework to characterize the multiscale nature of distributed systems.

To obtain a multiscale analysis of a complex distributed system, two methods may be applied. The first one is a bottom up approach, it studies at runtime real complex distributed system. Some recent works propose algorithms to study collaboration patterns of real complex systems. For example, in [5], the authors propose a community detection algorithm to find structures of communities to partition a complex system. These families of algorithms study the hierarchical structure of real complex systems. Due to the dynamic nature of complex systems, multiscale runtime analysis is unachievable. The second one is a top-down approach, it studies complex distributed systems at design time. New generations of approaches that allow developers of complex systems to work at a higher level of abstraction are needed [8]. Developing complex systems without such approaches increases the use of code-centric technologies, hinders developers from focusing on functional, non-functional and architectural needs, and requires herculean efforts. It produces hand-crafted systems, strongly coupled with technologies, that are neither maintainable nor upgradable. The use of MDE may help to describe systems at multiple levels of abstraction and from a variety of perspectives [17], [8]. MDE facilitates developers' work at design time, by providing specialized modeling languages, metamodels and code generation tools. Moreover, MDE can also be used at runtime to maintain a model of the running system. The interest of MDE is demonstrated in several research areas, as for context-aware pervasive systems [18], [19], runtime adaptation [20], or multi-cloud systems management [21]. Concerning the multiscale vision of complex systems, a multiscale UML profile for the deployment of complex systems is proposed in [22]. The model proposes three fixed scales: infrastructure, communication and deployed entities. In this approach, scales are rather views of a system. There is no distinction of levels in a given view —i.e., scales.

This study highlights the need for a top-down approach to study the multiscale nature of complex distributed systems. Model-driven approaches are interesting as they propose different levels of abstraction, and model transformations. It also identifies the lack of a shared multiscale taxonomy and scale-aware framework. We believe that the MuSCa framework fills these needs by helping system designers to build a multiscale vision of their complex systems. This framework is open : one

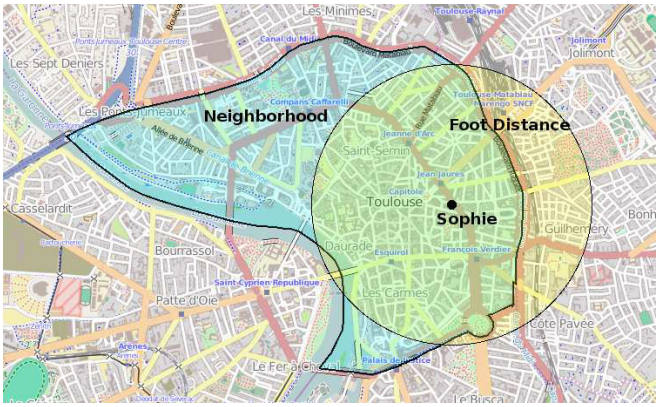

Fig. 7. Sophie's neighborhood and foot distance scale instances

[5]http://www.openstreetmap.org
[6]http://wiki.openstreetmap.org/wiki/Nominatim
[7]http://wiki.openstreetmap.org/wiki/JMapViewer

can add new viewpoints and associated dimensions, measures, scale sets and scales as needed.

VI. Conclusion

Multiscale distributed systems raise new kind of issues such as heterogeneity management, granularity variations, and distribution over the scales. This paper presented MuSCa, a framework to study multiscale nature of complex distributed systems and to provide them with scale-awareness capability.

The framework is based on a characterization process, which helps designers to study the multiscale nature of a given system. We have analyzed many scenarios and use cases for multiscale systems for the INCOME project. The characterization process enables the designer either to select—among existing ones—the viewpoints, dimensions and scales relevant for a given system, or to define new ones. For one characterization, MuSCa generates probes for runtime scale-awareness. Each characterization enables the framework to extend its multiscale taxonomy and its multiscale probes. Thus the framework learns and memorizes new viewpoints, dimensions and scales to be proposed for a later characterization.

MuSCa is helpful to build multiscale distributed systems that cope with some of the previously mentioned challenges. A good knowledge of the multiscale nature of a system contributes to choosing appropriate architectural patterns for each multiscale distributed system.

Currently, MuSCa is used the INCOME project. A DSL for software deployment was successfully designed using MuSCa. We also plan to use the scale-awareness capability to filter the distribution of the events in a distributed event based system.

Acknowledgments

This work is partly funded by the INCOME ANR project (ANR-11-INFR-009, 2012–2015) in which the following French partners are taking part: IRIT (Institut de Recherche en Informatique de Toulouse), Télécom SudParis, and ARTAL Technologies. The authors thank all the members of this project.

References

[1] M. Satyanarayanan, "Mobile computing: the next decade," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, no. 2, pp. 2–10, Aug. 2011. doi: 10.1145/2016598.2016600. [Online]. Available: http://dx.doi.org/10.1145/2016598.2016600

[2] M. van Steen, G. Pierre, and S. Voulgaris, "Challenges in very large distributed systems," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 59–66, 2012. doi: 10.1007/s13174-011-0043-x. [Online]. Available: http://dx.doi.org/10.1007/s13174-011-0043-x

[3] G. Blair and P. Grace, "Emergent middleware: Tackling the interoperability problem," *Internet Computing, IEEE*, vol. 16, no. 1, pp. 78–82, Jan. 2012. doi: 10.1109/MIC.2012.7. [Online]. Available: http://dx.doi.org/10.1109/MIC.2012.7

[4] D. Chavalarias *et al.*, "French roadmap for complex systems 2008-2009," Mar. 2009. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00392486

[5] P. Pons and M. Latapy, "Post-processing hierarchical community structures: Quality improvements and multi-scale view," *Theoretical Computer Science*, vol. 412, no. 8- 10, pp. 892–900, Mar. 2011. doi: 10.1016/j.tcs.2010.11.041. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2010.11.041

[6] M. Satyanarayanan, "Scalable, secure, and highly available distributed file access," *IEEE Computer*, vol. 23, no. 5, pp. 9–18, May 1990. doi: 10.1109/2.53351. [Online]. Available: http://dx.doi.org/10.1109/2.53351

[7] H. Sandhu and S. Zhou, "Cluster-based file replication in large-scale distributed systems," in *Proceedings of the ACM Sigmetrics Performance '92 Conference*, ser. SIGMETRICS '92/PERFORMANCE '92. New York, NY, USA: ACM, May 1992. doi: 10.1145/133057.133092. ISBN 0-89791-507-0 pp. 91–102. [Online]. Available: http://dx.doi.org/10.1145/133057.133092

[8] R. France and B. Rumpe, "Model-driven development of complex software: A research roadmap," in *2007 Future of Software Engineering*, ser. FOSE'07. Washington, DC, USA: IEEE Computer Society, 2007. doi: 10.1109/FOSE.2007.14. ISBN 0-7695-2829-5 pp. 37–54. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.14

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct. 2009. doi: 10.1109/MPRV.2009.82. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2009.82

[10] Object-Management-Group, "Software & Systems Process Engineering Metamodel (SPEM) v2.0," formal/2008-04-01, Apr. 2008.

[11] ISO/IEC/IEEE, "Systems and software engineering — architecture description," ISO/IEC/IEEE Joint Technical Committee, International Standard ISO/IEC/IEEE-42010:2011, Dec. 2011.

[12] J. Bézivin and O. Gerbé, "Towards a precise definition of the OMG/MDA framework," in *Proceedings. 16th Annual International Conference on Automated Software Engineering, 2001, (ASE 2001)*, Nov. 2001. doi: 10.1109/ASE.2001.989813. ISSN 1938-4300 pp. 273–280. [Online]. Available: http://dx.doi.org/10.1109/ASE.2001.989813

[13] R. Boujbel, S. Leriche, and J.-P. Arcangeli, "A DSL for multi-scale and autonomic software deployment," in *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, Oct. 2013, pp. 291–296.

[14] S. Machara Marquez, S. Chabridon, and C. Taconet, "Trust-based context contract models for the internet of things," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, Vietri Sul Mare, Italy, Dec. 2013. doi: 10.1109/UIC-ATC.2013.73 pp. 557–562. [Online]. Available: http://dx.doi.org/10.1109/UIC-ATC.2013.73

[15] M. Kessis, C. Roncancio, and A. Lefebvre, "DASIMA: A flexible management middleware in multi-scale contexts," in *Proc. Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09*, Apr. 2009. doi: 10.1109/ITNG.2009.338 pp. 1390–1396. [Online]. Available: http://dx.doi.org/10.1109/ITNG.2009.338

[16] J. P. Arcangeli *et al.*, "INCOME a multi-scale context management for the internet of things," in *Ambient Intelligence*, ser. Lecture Notes in Computer Science, F. Patern, B. Ruyter, P. Markopoulos, C. Santoro, E. Loenen, and K. Luyten, Eds. Springer Berlin Heidelberg, 2012, vol. 7683, pp. 338–347. ISBN 978-3-642-34897-6. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34898-3_25

[17] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *IEEE Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006. doi: 10.1109/MC.2006.58. [Online]. Available: http://dx.doi.org/10.1109/MC.2006.58

[18] E. Serral, P. Valderas, and V. Pelechano, "Towards the model driven development of context-aware pervasive systems," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 254–280, Apr. 2010. doi: 10.1016/j.pmcj.2009.07.006. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2009.07.006

[19] S. Chabridon, D. Conan, Z. Abid, and C. Taconet, "Building ubiquitous QoC-aware applications through model-driven software engineering," *Science of Computer Programming*, vol. 78, no. 10, pp. 1912–1929, Oct. 2013. doi: 10.1016/j.scico.2012.07.019. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2012.07.019

[20] G. Blair, N. Bencomo, and R. France, "Models@ run.time," *Computer*, vol. 42, no. 10, pp. 22–27, Oct. 2009. doi: 10.1109/MC.2009.326. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.326

[21] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems," in *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, ser. CLOUD '13. Washington, DC, USA: IEEE Computer Society, Jun. 2013. doi: 10.1109/CLOUD.2013.133. ISBN 978-0-7695-5028-2 pp. 887–894. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2013.133

[22] A. Gassara, I. Bouassida Rodriguez, and M. Jmaiel, "Towards a multi-scale modeling for architectural deployment based on bigraphs," in *Software Architecture*, ser. Lecture Notes in Computer Science, K. Drira, Ed. Springer Berlin Heidelberg, 2013, vol. 7957, pp. 122–129. ISBN 978-3-642-39030-2. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39031-9_11