The QoCIM framework : concepts and tools for quality of context management

Pierrick Marie, Thierry Desprats, Sophie Chabridon, Michelle Sibilla

Chapter 11
The QoCIM framework: concepts and tools for quality of context management

Pierrick MARIE, Thierry DESPRATS, Sophie CHABRIDON, Michelle SIBILLA

Abstract In the last decade, several works proposed their own list of quality of context (QoC) criteria. This article relates a comparative study of these successive propositions and shows that no consensus has been reached about the semantic and the comprehensiveness of QoC criteria. Facing this situation, the QoCIM meta-model offers a generic, computable and expressive solution to handle and exploit any QoC criterion within distributed context managers and context-aware applications. For validation purposes, the key modelling features of QoCIM are illustrated as well as the tool chain that provides developers with QoCIM based models editor and code generator. With the tool chain, developers are able to define and use their own QoC criteria within context and quality aware applications.

11.1 Introduction

The expansion of the Internet of Things (the extension of the Internet to objects of the real world), cloud computing, big data and mobile technologies foster the development of new ubiquitous, context- and situation-aware applications. These situations are computed from ambient data, profiles of users and information collected from heterogeneous and spatially distributed sources. Context-aware applications become more and more usual. These applications require a fine and efficient management of the quality of the context information (QoC) they rely on. QoC is related to any information that describes the quality of context data as stated by the seminal definition proposed by [4]. QoC specializes the general notion

Pierrick MARIE, Thierry DESPRATS, Michelle SIBILLA
IRIT UMR 5505 Université Paul SABATIER, 31062 TOULOUSE, France
e-mail: <Firstname>.<NAME>@irit.fr

Sophie CHABRIDON
Institut Mines-Télécom, CNRS UMR 5157 SAMOVAR, Télécom SudParis, 91011 Évry, France
e-mail: Sophie.Chabridon@telecom-sudparis.eu

of Quality of Information (QoI) for context information. A relevant behaviour of the QoC-aware applications strongly depends on the QoC they receive. However, according to the business objectives of these applications, some QoC criteria may appear more important than others. Sometimes the freshness criterion is sufficient, sometimes it is the precision criterion and other times both are necessary. A solution to handle this need is to use context managers. They support context information throughout their life cycle. The life cycle of a piece of context information begins at its creation by a sensor and ends at its consumption by a context-aware application. Between these two events, context data are aggregated, filtered, deduced or transformed many times [3]. These data are intrinsically incomplete and inaccurate [8]. A bad quality of context information could lead to wrong decisions and irrelevant reactions. That is why context managers must take into account QoC at each step of the context information life cycle. This challenge logically remains in the case of the next generation of multi-scale distributed context managers.

The extension of the scope of context managers from local ambient environments to the Internet of Things (IoT) leads to a spatio-temporal decoupling between context providers like raw data producers close to RFID readers or sensors networks, and context consumers that are context-aware applications running, for example, on mobile devices close to users. This kind of middleware must be deployed over various devices or servers, spread across various networks or clouds, and we name them *Multiscale Distributed Context Managers* (MDCM). Several solutions have already been proposed. In 2007, the AWARENESS project [17] proposed a middleware to manage context information and offered a way to manipulate the QoC. In 2009, the COSMOS project [1] proposed mechanisms for the efficient management of QoC for ambient intelligence. In 2011 [9] proposed a DSL (MLContext) and a process to easily develop context-quality aware applications. With the DSL developers are able to create new context and QoC aware applications. MLContext offers the benefits of considering the QoC in terms of guaranties for the producers of context and in terms of QoC requirements for the consumers of context. Finally, one of the objectives of the INCOME project [2], started in 2012, is to design solutions able to handle QoC as well as to preserve privacy within a new MDCM.

We intend to provide future context managers with a *generic*, *computable* and *expressive* way to manipulate and exploit QoC simply and efficiently. *Generic*, because our solution has to model complex and heterogeneous QoC criteria. *Computable*, because the estimation of the quality level of context information is based on treatments and operations on QoC criteria. Lastly, *expressive*, because context-aware applications must be able to express their QoC requirements to different context managers.

This paper is organized as follows. Section 11.2 compares the lists of QoC criteria that have been proposed over the last decade. Section 11.3 illustrates with a fictional scenario what kind of services the new generation of context manager have to fulfil. After having found, in Section 11.2, no standard list of criteria to measure QoC and illustrate, in Section 11.3, the necessity to handle the QoC within MDMC, we

propose the *Quality of Context Information Meta-model* (QoCIM) in Section 11.4. It brings a *generic*, *computable* and *expressive* solution to manipulate and manage QoC. The modelling key points of QoCIM are illustrated in Section 11.5. Finally, Section 11.6 presents the software tool chain we have built to produce and to manage libraries of QoCIM-based QoC criteria models and Section 11.7 concludes this paper.

11.2 Comparative study of existing QoC criteria lists

We study in this section the existing works about QoC measurement. Many authors have already established their own list of QoC criteria to measure QoC. We first enumerate the main proposals published over the last decade, and finally we compare the proposed criteria with regard to their semantics. The study highlights the existing variations in terms of name and meaning of QoC criteria. Different authors define a same meaning but associate it with a different denomination. On the contrary, a same denomination defined by different authors may correspond to different meanings.

11.2.1 Overview of QoC criteria lists

Buchholz et al., 2003 [4] proposed the first list of QoC criteria for context-aware services. This list is composed of five criteria : *precision*, *probability of correctness*, *trust-worthiness*, *resolution* and *up-to-dateness*. All of them are defined through a textual description. No computation method is formulated for their estimation, but the authors provide examples to illustrate each of them.

Kim and Lee, 2006 [11] proposed a new list of QoC criteria built by confronting Buchholz et al.'s QoC criteria to generic criteria to measure quality. The authors provided five criteria associated to a definition from the point of view of the end-users of the context information. The end-user is the last entity which consumes context information. The proposed criteria are *accuracy*, *completeness*, *representation consistency*, *access security* and *up-to-dateness*. Then, they defined a mathematical formula to estimate the value of their first two criteria : *accuracy* and *completeness*.

Sheikh et al., 2007 [17] formulated their own list of QoC criteria for the AWARENESS project. These criteria are *precision*, *freshness*, *temporal resolution*, *spatial resolution*, and *probability of correctness*. Although these criteria are textually described, no method is provided to estimate their value. Like Buchholz et al., Sheikh et al. gave examples to illustrate the definitions of their criteria. The descriptions of the criteria adopt successively the points of view of the consumer and of the producer of the context information. Producers are entities that create context information such as sensors, while consumers are context-aware applications.

Filho, 2010 [7] studied the lists of QoC criteria that had been previously listed in [4], [11] and [17] and proposed a new list of QoC criteria for the access control security domain. Filho redefined *up-to-dateness*, *sensitiveness*, *access security*, *completeness*, *precision* and *resolution* criteria. For each criterion, Filho offered an example to illustrate the notion which is measured. He also provided a mathematical formula or a sample Java program that he used to estimate these criteria.

Neisse, 2012 [14] suggested adapting the ISO standard used in metrology to define QoC criteria. He established that the concepts of *accuracy* and *precision* used as QoC criteria are just an approximative definition of the precision criterion used in metrology. In the same way, Neisse estimated that the concepts of *spatial resolution* and *temporal resolution* defined by [17] are just a redefinition of the ISO standard of precision applied to spatial and temporal information. Neisse suggested measuring the QoC with only two criteria: the *age* and the *precision* of the context information. The *age* is the elapsed time since the production of the information. The *precision* criterion applies the ISO standard of measurement precision to other kinds of information depending on the needs of the application. So, this *precision* criterion could be applied to the location of the source of the information, for example.

Manzoor et al., 2012 [12] offered the most complete list of QoC criteria. They defined seven high level QoC criteria which depend on other lower level QoC criteria. For each of these high level QoC criteria, the authors associates a mathematical formula. The proposed criteria are *reliability*, *timeliness*, *completeness*, *significance*, *usability*, *access right*, *representation consistency*. The definition of some criteria, like the significance, adopts the point of view of the context producer. The significance "indicates the worth or the preciousness of context information in a specific situation" where the context producer is a sensor. Whereas the definition of other criteria adopts the point of view of the context consumer. For example, the criterion representation consistency is computed with information coming from requirements expressed by the context-aware applications in terms of QoC. The criterion "depends upon the amount of effort that is needed to transform that context object according to the data model presented by context consumer".

11.2.2 Discussion

The study of the semantics of the QoC criteria listed above shows some divergences. The same name of a criterion appears in several lists with a different meaning. Conversely, a same meaning appears in several lists with different denominations. There are also meanings associated with denominations that appear only once into all the lists. Table 11.1 groups together the studied criteria by author and highlights the differences that exist between all of these criteria.

The different lists of QoC criteria are represented vertically. The name and the year of the first author of each list are mentioned on the first line and are sorted chronologically. We associate a number to each criterion, which is indicated in

Table 11.1 Comparison of different lists of QoC criteria

		Buchholz et al. 2003 [4]	Kim and Lee 2006 [11]	Sheikh et al. 2007 [17]	Filho 2010 [7]	Manzoor et al. 2012 [12]	Neisse 2012 [14]
1	Probability context is free of errors	**Correctness**	Accuracy		**Precision**	Accuracy	
2	Max. distance to get context					*Sensor range*	
3	Location of the real world entity					*Entity location*	
4	Location of the sensor					*Sensor location*	
5	Time between production of contexts			*Temporal resolution*	✓	*Time period*	
6	Date of collection of context	✓	✓	✓	✓	*Measurement time*	*Timestamps*
7	Granularity of location			*Spatial resolution*	**Resolution**		
8	Rate the confidence of the provider	*Trust worthiness*					
9	Critical value of context					*Significance*	
10	Closeness, Repeatability of measurements (ISO)						**Precision**
11	Granularity (detail level) of context	**Precision**		**Precision**	*Sensitiveness*	*Usability*	
12	Context consumer have access to context		✓			*Access right*	
13	Context transfers restricted, secured		Access security (11)		Access security		
14	Format respects consumer needs		Consistency			Consistency	
15	Validity of context based on freshness	**Up to dateness** (6)	**Up to dateness** (6)	*Freshness* (6)	**Up to dateness** (5, 6)	*Timeliness* (5, 6)	
16	All aspects of entity are available	**Resolution**	Completeness		Completeness (15)	Completeness	
17	Belief in the correctness of context			**Correctness**		*Reliability* (1, 2, 3, 4)	

Meaning	Meaning used by all authors
Name	Criterion (name + meaning) only defined by one author
Name	Name only defined by one author
Name	Name defined by different authors with different meanings
Name	Name defined by different authors with the same meaning
Name (X)	The definition of this criterion depends on the criterion number X
✓	Criterion not defined by author but another criterion depends on it

the first column of the table. The second column summarizes the meaning of each criterion. The cells of the table which contain a name correspond to criteria proposed by the authors. An empty cell indicates that the authors did not propose the criterion on this line. A cell with a check-mark represents a criterion implicitly used by the corresponding author but not clearly defined in its list of QoC criteria. Grey cells represent criteria defined by only one author. The lightgrey color indicates that there is one common meaning used by all authors. The criteria written in italic are names used only once. The criteria written in bold are names used by at least two different authors with different meanings. Some names of criterion are followed by numbers. For example, on line 17, the reliability criterion defined by Manzoor [12] is followed by the numbers 1, 2, 3 and 4. These numbers reference the numbers in the first column and indicate that this criterion is composed of other criteria. For this example, the criterion is computed using the first four criteria listed in this table.

Lastly, QoC criteria are sorted in the table by following a specific order. Criteria extracted directly from raw sensor data and which do not need computation or statistical analysis are placed on the top of the table. Whereas criteria at the bottom of the table require historical analysis or data from many sensors to be estimated. The more a criterion requires computations and data, the lower it is placed in the table. Manzoor [12] classifies criteria into two categories, objective and subjective criteria; an objective criterion does not depend on the final application whereas a subjective criterion depends on the purpose of the final application. Table 11.1 orders criteria as a function of the effort that is required to estimate them.

Table 11.1 highlights that there is no consensus about which QoC criteria have to be used to measure the QoC of context information. This supports the idea of [3] indicating that a consensus about the definition of a common list of QoC criteria is still an open problem. Moreover the table provides a way to compare different lists of QoC criteria. This makes it possible to compare new specific lists between them. Indeed, with the development of context-aware applications, if a new high level criteria appears, Table 11.1 offers a method to classify lists of QoC criteria relatively to one another.

Despite the plethora of QoC criteria, MDCM still have to handle the QoC all along the life cycle of context information. Using MDCMs implies that it is no longer possible to establish a kind of “one-to-one QoC-based contract” between context data producers and respective consumers. Nevertheless, QoC requirements subsist and MDCMs have to match the quality of the context information that is delivered to a consumer with its expectations. Symmetrically, MDCMs have to know about the guarantees that a context data producer claims about some related QoC criteria. Lastly, because one of the main functionalities of context managers is to apply some processing to context information (aggregation, inference and so on), they also have to tackle the QoC during the execution of these operations. Consequently, MDCMs should be extensible by enabling the definition of any QoC criterion including their associated computation algorithm. A solution to supply QoC management within MDCM is to use a common way to model QoC criteria, to compute the value of the QoC and to express requirements and guarantees. The next

Section presents an example to illustrate the services that new generation of context manager have to fulfil.

11.3 Scenario

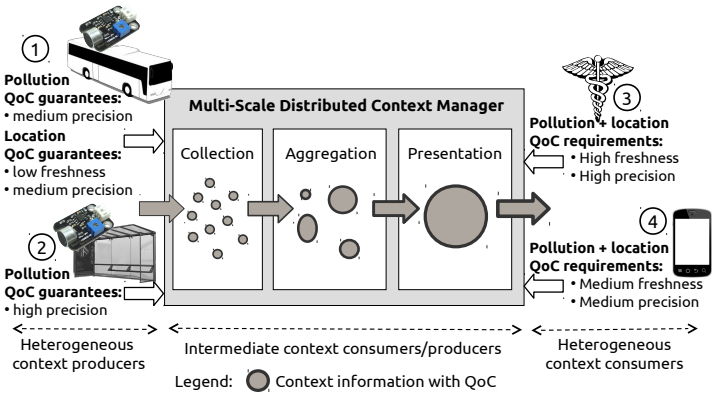

Fig. 11.1 QoC for the pollution measurement scenario

This Section describes a fictional scenario inspired from an existing concern: the urban pollution. We plan to develop the scenario in the future to experiment our solution. The future experimentations may be based on real pollution measurements realised with sensors networks or with simulated and random measurements. In our scenario, a city installs on its public transportation buses a pollution sensor and a location sensor like a GPS to inform users about the most polluted streets. The city also installs more sophisticated pollution sensors in each bus station. To improve the performance and increase the utility of the offered services, an embedded software is associated to the pollution and location measurement sensors. The software uses both the precision, criterion number 10 defined by [14], and the freshness, criterion number 15 defined by [17], to qualify the location of the buses, and it only uses the precision to qualify the pollution measured on the buses and the bus stations. The MDCM provides different QoC-aware applications with context information about the pollution measured by the buses and the bus stations in the city. The MDCM also provides information about the quality of the context information. The QoC is presented to the QoC-aware applications as meta-data associated to the context information.

In this example, as shown on Figure 11.1, the buses (1) are committed to providing their location with at least low freshness and medium precision and their pollution measurement with at least medium freshness. The bus stations (2) are committed to providing their pollution measurement with at least high precision.

Context-aware applications will receive information concerning the location of the most polluted streets. The health care application (3) requires context information with at least high freshness and high precision. The application is used by asthmatic people to avoid the most polluted streets when they walk in the city. A general mass-market mobile application (4) requires context information with at least medium freshness and medium precision. This application is used by healthy people to get news about the pollution in the city.

In the case where the health care application does not receive context information with the expected QoC, the application will stop its services and display a message to indicate it does not have enough information to provide its services. Indeed, the health care application is critical for asthmatic people. It is preferable to not provide any service instead of providing erroneous indications. For the second application, if it does not receive the QoC that it expected, the application will continue to provide its services but a warning will be displayed. It indicates the users have to momentarily decrease their confidence into the application. This architecture highlights the necessity to use a common model of QoC criteria to: (*i*) measure the QoC of a context information, (*ii*) express QoC requirements and guarantees, (*iii*) help context manager to deal with the QoC. The next Section presents QoCIM, our solution to answer to these problems.

11.4 QoCIM : A new QoC meta-model

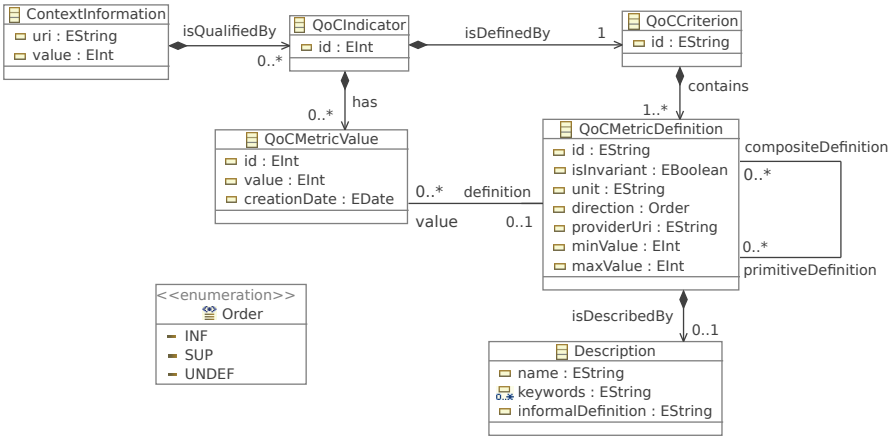

Fig. 11.2 QoCIM : QoC Information Model

QoCIM is our proposed meta-model for designing and representing QoC. According to [16] a meta-model "is used to refer to a model of some kind of meta-data" and meta-data "is used to refer to data whose purpose is to describe

other data". As described in the next section, we use QoCIM to build other models of QoC indicators, that is why we consider QoCIM as a meta-model. QoCIM is not dependent on any QoC criterion. It offers a unified solution to model, at design time, heterogeneous QoC criteria. The key modelling points of QoCIM are inspired from interesting concepts or modelling patterns used in several existing models studied in [13]. Thus, models based on QoCIM could be used, at runtime, by both MDCM and QoC-aware applications, for the dynamic handling of QoC. This section briefly describes the QoCIM meta-model introduced in [13].

11.4.1 Presentation of QoCIM

Figure 11.2 presents the QoCIM meta-model. QoCIM qualifies context information represented with the class `ContextInformation`. An indicator is represented with the class `QoCIndicator`, it contains the quality of context information and is defined by one criterion, with the class `QoCCriterion`. Indicators and criteria are identified uniquely with the attribute `id`. At runtime, a valuation of the QoC is available with instances of the class `QoCMetricValue`, identified with the attribute `id`. Its `value` attribute provides a valuation of the QoC. The date of creation of a value is contained into the attribute `creationDate`. The attributes of the class `QoCMetricDefinition` define the production of instances of `QoCMetricValues`:

- `isInvariant` indicates whether the produced value is a constant, neither editable, nor dynamically computed.
- `unit` represents the unit of the produced value. It could be one of the units of the International System.
- `direction` compares different `QoCMetricValues` based on their attribute `value` from the point of view of the consumer of context information. The possible values of this attribute are INF, SUP and $UNDEF$:
  - INF means that a high `value` induces a better QoC level. For example, the freshness, or age, of a piece of context information is usually computed with the following formula:
    $freshness = current\ date - date\ of\ the\ production\ of\ the\ context.$
    The result of this operation increases with the time whereas the quality of the information decreases.
  - SUP means that a high `value` induces a worse QoC level. For example, the spatial reliability of a piece of context information, that indicates how much we can trust a sensor according to its distance to the observed entity, could be computed with the following formula:
    $$spatial\ reliability = 1 - \frac{distance\ between\ sensor\ and\ observed\ entity}{maximum\ distance\ for\ sensor\ to\ get\ context}$$
    If the sensor is close to the context, the result of this operation and the quality of context will be greater than if the sensor is far to the context.
  - $UNDEF$ is used when neither INF nor SUP can be expressed.

- `providerUri` identifies the resource that provides the `QoCMetricValue`. This attribute brings a way to filter the QoC based on the entity which computed it at runtime.
- `minValue` and `maxValue` respectively define the minimum and the maximum allowed value of the attribute `value` of the class `QoCMetricValue`.

The class `Description` brings semantics for the class `QoCMetricDefinition`. The attribute name contains the `name` of the description. The attribute `keywords` is a list of keywords. Finally, the attribute `informalDefinition` is a text that informally describes the `QoCMetricDefinition`. For the purpose of building composite indicators, the recursive association set on the class `QoCMetricDefinition` supports the ability to model and use a resulting indicator based on other indicators. Therefore, QoCIM authorizes `QoCMetricDefinition` depending on other classes `QoCMetricDefinition`.

11.4.2 Discussion

The analysis of existing models presented in [13] highlights interesting concepts of modelling patterns used in QoCIM. The first concept comes from the meta-model of the IoT-A [10] project. It proposes to associate meta-data with context information. QoCIM also uses this technique with the classes `ContextInformation` and `QoCIndicator`. Like the DMTF CIM metrics model [6], QoCIM separates the metrics definition, `QoCMetricDefinition`, from the metrics value, `QoCMetricValue`. QoCIM reuses a few attributes of the Object Management Group (OMG) QoS meta-model [15] like `isInvariant`, `direction` and `unit`. QoCIM adds other attributes, like `providerUri`, and the class `Description` which are not specified in the OMG QoS meta-model. The DMTF CIM metrics model and the OMG QoS meta-model build higher level complex definitions of metrics based on other definitions of metrics. With the same objective, QoCIM also gives designers of context-aware applications the ability to specify new composite QoC indicators thanks to the recursive link set on the class `QoCMetricDefinition`.

11.5 The key modelling features of QoCIM

The following paragraphs describe Figures 11.3 and 11.4 and illustrate the six main modelling features of QoCIM. The figures are two UML[1] class diagrams based on QoCIM. Figures 11.3 and 11.4 have been built with a “QoCIM models editor” that we present in the next Section.

[1] Unified Modeling Language: www.uml.org

Feature 1: Qualifying information with several QoC indicators
With the example of two pieces of context information, the pollution measurement and the location, Figure 11.3 exhibits how QoCIM qualifies context information. In this example, the location information is qualified with two QoC indicators, the freshness and the precision. Freshness corresponds to the fifteenth definition referenced in Table 11.1. According to [17], freshness "is the time that elapses between the determination of context information and its delivery to a requester". Precision corresponds to the tenth definition referenced in Table 11.1. According to [14] and [5], precision "is defined as how close together or how repeatable the results from a measurement are". Qualifying a piece of context information with different QoC indicators allows to analyse the information from different points of view. It is thus possible to get a complete opinion of the real quality of the information and provides QoC-aware applications with all they need to deliver relevant services to end-users.

Feature 2: Reusing a QoC indicator
Offering a way to reuse already defined QoC indicators eases the development of QoC-aware applications. This feature is supported by QoCIM. It enables developers of QoC-aware applications to reuse already defined QoC indicators for their needs. As illustrated in Figure 11.3, the precision indicator qualifies two pieces of context information at the same time, the pollution measurement and the location. With a collection of QoC indicators, developers indeed just have to pick-up what indicators they need for their QoC-aware applications.

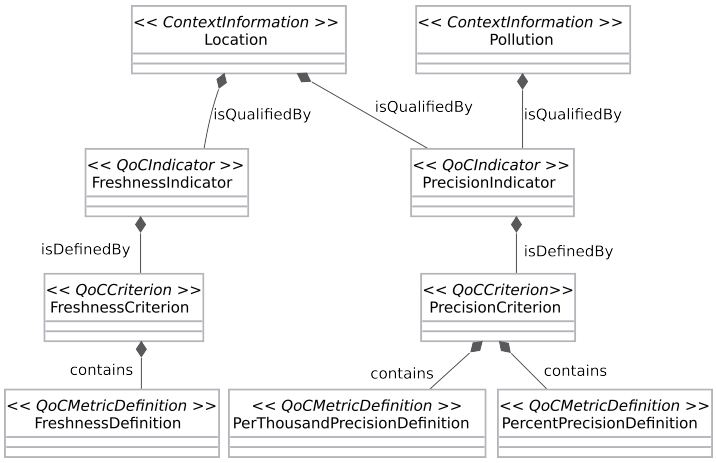

Fig. 11.3 QoCIM based model of multiple QoC criteria definitions

Feature 3: Defining a QoC criterion with one definition
In Figure 11.3, the definition of freshness provided by [17] is evaluated in a single way. Only one definition, `FreshnessDefinition`, is used to measure the freshness of the location context information. This is the simplest way to define a QoC

criterion with QoCIM, with only one definition per criterion. The next paragraph describes a more complex way to define a criterion.

Feature 4: Defining a QoC criterion with multiple definitions

In Figure 11.3, the definition of precision provided by [5] can be evaluated in multiple ways. In our example, we illustrate this plurality by providing two different definitions associated to the precision criterion. One definition expresses the precision in percent while the other definition expresses the precision in per thousand. They still have the same semantics but their implementation will differ. Providing multiple definitions for a same criterion allows different sensors with different capabilities to choose which definition is more appropriate to qualify their measurements according to their properties. For example, in the scenario described in Section 11.3, sensors placed on bus stations are more sophisticated than sensors placed on buses. A consequence of this difference could be a different definition used to compute the value of the precision. The precision of the measurements made with the sensors placed on the buses will be express in per-hundred while the precision of the measurements made with the sensors placed on the bus stations will be express in per-thousand.

Feature 5: Composing multiple definitions

Figure 11.4 presents the definition of a composite indicator. The composite indicator depends on the classes `PercentPrecisionDefinition` and `FreshnessDefinition`. These classes are the implementation of the freshness and the precision indicator presented previously adapted from the class `QocMetricDefinition` described in Section 11.4. The `id` of the composite indicator is 18 because it could be classified into Table 11.1 as a new indicator, that is to say the eighteenth indicator. The value of the `id` of the class `CompositeCriterion` is "15.1 – 10.1". This value corresponds to the concatenation of the value of the `id` of the classes `FreshnessDefinition`, which is "15.1", and `PercentPrecisionDefinition`, which is "10.1". The value of the attribute `id` of the class `CompositeDefinition` is "18.1", because the `CompositeDefinition` is the first definition of the eighteenth indicator. As for the precision indicator, the value of the attributes `direction` of the class `CompositeDefinition` is *SUP*. It means that the more the value of this indicator increases, the more the quality of the context information increases. The computation of these values depends on the combined evaluation of the two primitive indicators, precision and freshness.

Feature 6: Producing discrete values

In Figure 11.4, the high level indicator may take three different `QoCMetricValues`: `HighValue`, `MediumValue` and `LowValue`. These `QoCMetricValues` are respectively associated to a default value: 1, 2 or 3. The computation of these values are specified with the Object Constraint Language (OCL)[2]. As an example, listing 11.1 shows the mandatory constraints to produce a `HighValue`. With few OCL constraints, QoCIM allows to create discrete values based on two continuous values, in this example: the values of the precision and freshness.

[2] Object Constraint Language: www.omg.org/spec/OCL

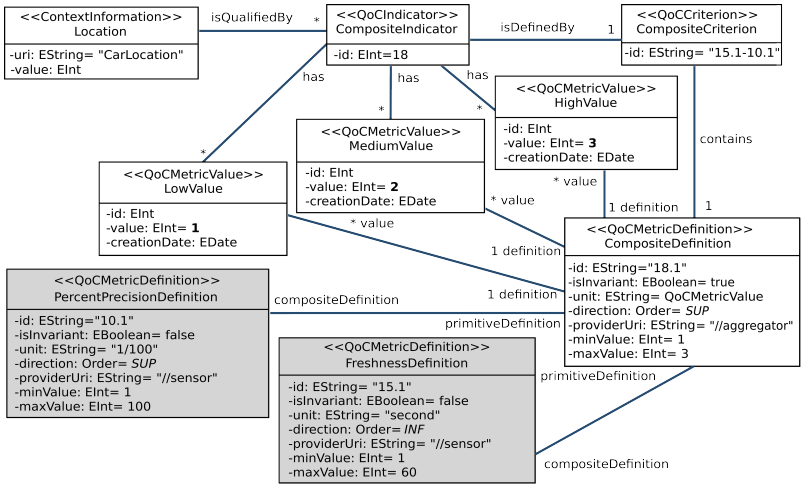

Fig. 11.4 QoCIM based model of a QoC composite indicator

```
context CompositeDefinition::value(): HighValue
  pre: self.PerCentPrecisionDefinition.QoCMetricValue.value >=
    85 % self.PrecisionDefinition.maxValue
  pre: self.FreshnessDefinition.QoCMetricValue.value <=
    15 % self.FreshnessDefinition.maxValue
```

Listing 11.1 OCL constraints to define HighValue for the composite indicators

Figures 11.3 and 11.4 illustrate the six most important modelling features of QoCIM. However, manipulating a meta-model to build new models without dedicated tools rapidly becomes complex and error prone. If the developers of QoC-aware applications cannot easily handle models based on QoCIM, they will prefer to build their own QoC solution to the detriment of the interoperability with MDCMs or with the other QoC-aware applications.

11.6 QoCIM software engineering tool chain

This section presents the software tool chain that we developed to build a library of QoCIM-based QoC indicator models. The purpose of the tool chain is a first step to easily defining new primitive or composite models of QoC indicators. As examples, Figures 11.3 and 11.4 have been produced with the tool. In a second step, the tool chain can automatically generate the source code corresponding to the models previously defined. Figure 11.5 illustrates the two steps of this process to build a library with the source code of QoC indicator models.

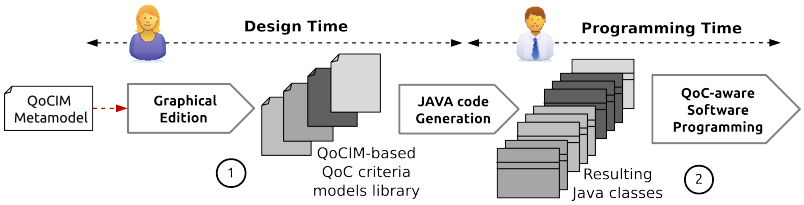

Fig. 11.5 Software tool chain process

Step 1, designing new QoCIM-based QoC indicator models

With the graphical tool, the designers define new QoC indicators based on QoCIM. The graphical tool is a dedicated software that aims to graphically produce new UML class diagrams of QoC indicators. It is possible to edit the indicators models by adding new definitions, new descriptions or new discrete values. With the tool, it is also possible to define new composite indicators depending on other already defined indicators. The designers are able to create and handle their own library of QoC indicator models. The tool offers a unified way to create QoCIM-based indicators model and manage them within libraries. The graphical models handled by the editor are stored into XML files, enabling sharing and maintaining a library of QoC indicators. Then, the designers may share their library of QoC indicators models with others by using an online repository of models, for example. The other designers just have to use the tool to pick-up, modify or complete the models according to their needs. Using a graphical tool is a solution to easily handle a collection of QoC indicator models.

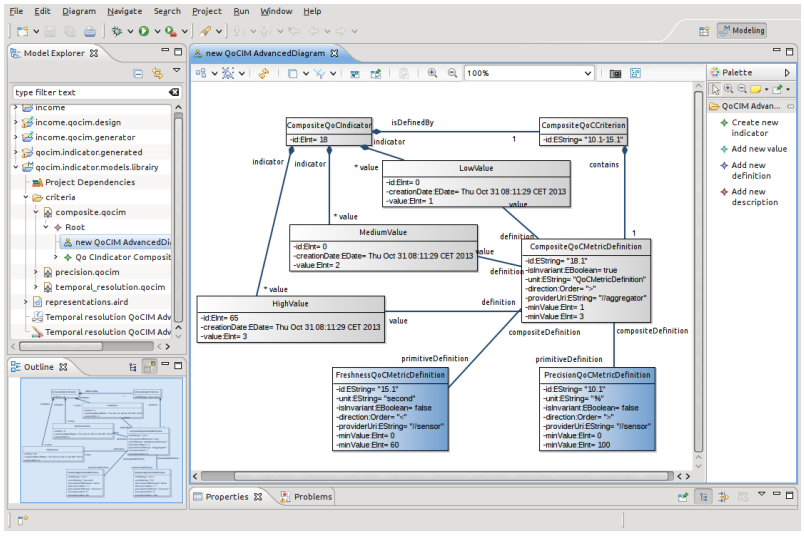

Fig. 11.6 Screenshot of the part of the tool dedicated to edit models

Step 2, generating the source code of QoC indicators

At the end of the first step, designers will possess a set of QoC indicator diagrams. At programming time, developers choose from the set of QoC indicator diagrams what they desire for their QoC-aware applications and with the tool generate the code corresponding to the indicators. The generated code will handle the QoC within QoC-aware applications or MDCMs. Currently, the software is able to generate Java code but, for future works, the QoC indicator diagrams could be translated into many different programming languages. We use the strength of the Object-oriented paradigm supported by Java to manipulate the generated classes through the factory method pattern. This provides developers with an easy way to manipulate the classes of the models within their applications. Then, computing the right value of QoC according to the definition of the indicator is based on the delegation pattern and isolated into an empty method. Once the code corresponding to the QoC indicators is produced, the developers just have to complete this empty method to evaluate the QoC within QoC-aware applications.

Figure 11.6 and 11.7 are two screenshots of the tool. Figure 11.6 shows the graphical editor used to design new QoC indicators diagrams. Figure 11.7 represents a sample of the Java code generated with the editor from a QoC indicator diagram.

The QoC indicator manipulation tool that we developed is built with the Open Source Obeo technology[3]. It is a software based on the Eclipse Modelling Framework (EMF) technology[4]. With the Obeo software, it is possible to easily create and configure new software dedicated to the manipulation of models based on a meta-model. The resulting software that we configured is able to graphically create new models based on QoCIM. The configuration files that we wrote are specific to the Obeo technology. We also wrote template files that are used by our graphical editor to generate source code corresponding to models based on QoCIM. The editor analyses the models and applies instructions contained in the template to generate new text files. We configured the template files to generate new Java classes.

Four steps are necessary to develop a QoC-aware application with the DSL and the process proposed by [9]: (1) describe the observed context entities, (2) express the capabilities of the context sources, (3) specify the QoC attributes of the providers of context, (4) identify the QoC requirements of the applications. MLContext handles the QoC in the steps 3 and 4 but with “the most commonly used quality attributes in the literature” and a key-values system. Our solution is focused on QoC management and is able to defining and handling a large collection of QoC criteria. That is why, to extend the usability of QoCIM and improve its expressiveness to develop QoC-aware applications, future works could consist to integrate QoCIM within MLContext to express QoC requirements and guaranties with a generic model of QoC criteria instead of using a pre-defined list of criteria.

[3] Obeo Designer v6.2: www.obeodesigner.com/download

[4] Eclipse Modelling Framework: www.eclipse.org/modeling/emf

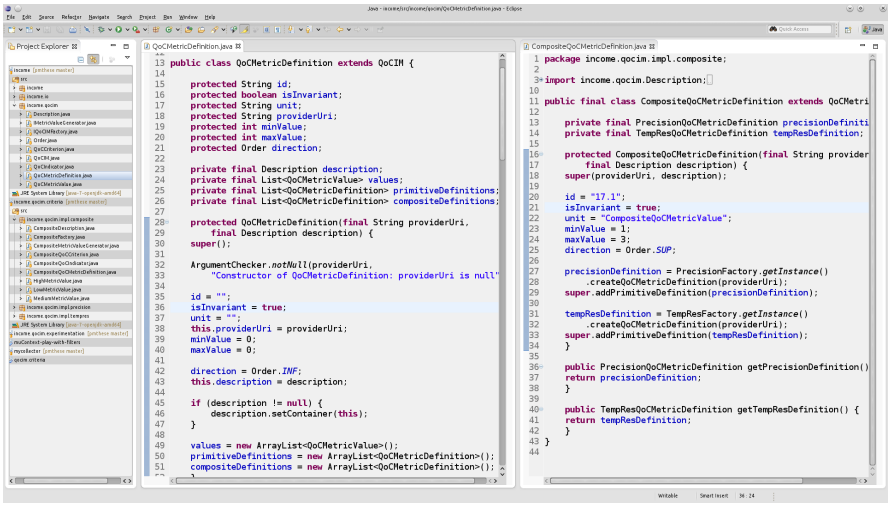

Fig. 11.7 Screenshot of the part of the tool dedicated to the code generation

11.7 Conclusion and perspectives

In the last decade, several works have addressed QoC modelling and management. This article presents the result of our analysis of some of the QoC criteria lists proposed by different authors. The analysis explicitly demonstrates the existence of divergences and concludes on the difficulty to converge to a unique and exhaustive QoC criteria list. Facing this situation, we propose the QoCIM meta-model. QoCIM is dedicated to exploit and manipulate any QoC indicator within MDCM and QoC-aware applications. This article introduces the informational core of QoCIM. Then, it presents the key modelling elements of QoCIM that ease the definition of QoC indicators and the qualification of context information. Because reasoning with a meta-model is not convenient, the article describes a graphical model editor that helps developers to build and integrate QoC models within their QoC-aware applications. We are currently working on identifying and defining QoC processing functions that occur all along the life cycle of context information. The purpose is to find the potential relationships that exist between the functions of context information transformation (fusion, aggregation, interpretation, inference) and QoC processing functions. Identifying QoC processing functions will allow us to build graphs, such as coloured Petri nets, to visualize and formalize the construction of high level context information delivered to context consumers.

Acknowledgment

This work is part of the French National Research Agency (ANR) INCOME project (ANR-11-INFR-009, 2012-2015, http://anr-income.fr).

References

1. Abid, Z., Chabridon, S., Conan, D.: A Framework for Quality of Context Management. In: First Int. Workshop on Quality of context, Lecture Notes in Computer Science (2009)
2. Arcangeli, J.P., et al.: INCOME - Multi-scale Context Management for the Internet of Things. In: Conf. on Ambient Intelligence (AmI), LNCS 7683. Pisa, Italy (2012)
3. Bellavista, P., Corradi, A., Fanelli, M., Foschini, L.: A Survey of Context Data Distribution for Mobile Ubiquitous Systems. ACM Computing Surveys (2012)
4. Buchholz, T., Kupper, A., Schiffers, M.: Quality of Context Information: What it is and why we Need it. In: 10th Int. Workshop of the HP OpenView University Association (2003)
5. CEI and ISO: International vocabulary of basic and general terms in metrology (VIM) (2004)
6. Distributed Management Task Force: Base Metric Profile (2009)
7. Filho, J.B.: A family of context-based access control models for pervasive environments. Ph.D. thesis, University of Grenoble Joseph Fourier (2010)
8. Henricksen, K., Indulska, J.: Modelling and using imperfect context information. In: Proc. 1st PerCom Workshop CoMoRea (2004)
9. Hoyos, J., Preuveneers, D., García-Molina, J., Berbers, Y.: A dsl for context quality modeling in context-aware applications. In: Ambient Intelligence - Software and Applications, Advances in Intelligent and Soft Computing (2011)
10. Internet of Thing Architecture Project: Deliverable 1.3 (2012)
11. Kim, Y., Lee, K.: A quality measurement method of context information in ubiquitous environments. In: Proceedings of the International Conference on Hybrid Information Technology (2006)
12. Manzoor, A., Truong, H.L., Dustdar, S.: Quality of context models and applications for context-aware systems in pervasive environments. Knowledge Engineering Review Special Issue on Web and Mobile Information Services (2012)
13. Marie, P., Desprats, T., Chabridon, S., Sibilla, M.: QoCIM: a meta-model for quality of context. In: CONTEXT'13 : Eighth International and Interdisciplinary Conference on Modeling and Using Context (2013)
14. Neisse, R.: Trust and privacy management support for context-aware service platforms. Ph.D. thesis, University of Twente, Enschede (2012)
15. Object Management Group: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification (2008)
16. Open Geospatial Consortium: Meta Object Facility (MOF) Specification (2005)
17. Sheikh, K., Wegdam, M., Van Sinderen, M.: Middleware Support for Quality of Context in Pervasive Context-Aware Systems. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (2007)