Model-Driven Engineering of Software Solutions for QoS Management in Real-Time DBMS

Salwa M'barek[a], Leila Baccouche[b], and Henda Ben Ghezala[c]
RIADI Laboratory
ENSI, Manouba University
Tunis, Tunisia
e-mail: [a]salwa.mbarek@gmail.com, [b]leila.baccouche@insat.rnu.tn, [c]hhbg.hhbg@gmail.com

***Abstract*—Real-Time applications handling big volumes of Real-Time data with time constraints, such as web-based multimedia applications or Vehicular Cyber-physical Systems, often lead to unpredictable overload problems in Real-Time DataBase Management Systems (RTDBMS) managing them. This is due to frequent Real-Time user transactions, requesting access to Real-Time data, which are characterized by unknown arrival times, unknown workloads and time constraints. In the literature, many software solutions with Quality of Service (QoS) management are proposed to resolve these problems. Although effective, they remain application-dependent regarding Real-Time data and transactions models and QoS requirements. Moreover, no formal or semi-formal models of these solutions or tools are proposed to design them. Therefore, it is not possible to reuse such solutions for Real-Time applications with specific QoS requirements and needing other data and transactions models. To address this issue, we propose a Model Driven Engineering based framework for modeling QoS management solutions in RTDBMS and reusing formal models of well-known solutions. The framework provides a tool and strategic methodology to help users achieve their goals through several strategies that fit their requirements.**

Keywords- Model Driven Engineering; QoS management; Real-Time DBMS; model transformations; reuse.

I. INTRODUCTION

Real-Time DataBase Management Systems (RTDBMS) are suitable to Real-Time Applications (RTA) handling a large volume of Real-Time data, which have validity periods and must be updated periodically to reflect the state of the application environment. They are able to manage Real-Time transactions, which are time-constrained (e.g., having deadlines) and frequently request access to Real-Time data [18]. Examples include the Vehicular Cyber-Physical Systems (VCPS) that must collect and handle a large volume of Real-Time data about vehicles and road traffic for ensuring road safety and providing various data services within accurate time deadlines [14]. There are also included web-based multimedia applications, such as video on demand, which manage large amounts of data and must respect the time constraints when transmitting video packets [12].

A RTDBMS executes periodic *update transactions*, which refresh Real-Time data to preserve the logical and temporal consistency of the Real-Time database. In addition, it manages transactions from users reading the Real-Time data, called *user transactions* and must meet their deadlines [8].

The workload of *update transactions* is known in advance, while the *user transactions* have unknown workloads and unpredictable arrival times [18]. Hence, the RTDBMS can face unpredictable overload periods and be no longer able to satisfy both types of transactions, which lead many Real-Time transactions to miss their deadlines. Thus, the RTDBMS stability may not be guaranteed [8].

In the literature, many QoS-aware solutions that we call QoS Management Solutions (QMS) aim to address this problem, such as the solutions proposed in [2][3][11][12]. The QoS guarantee is based on QoS requirements specified by the database administrator (DBA). Most QMS combine the Feedback Control Scheduling Architecture (FCSA) [16] with imprecise computation techniques [10], which allow graceful QoS degradation during transient overloads.

Although effective, these solutions are dependent on the Real-Time data and transactions models and requirements of the RTA. Moreover, we did not find in the literature formal or semi-formal models of these solutions or tools to model them. Therefore, it is not possible to reuse existing solutions for a RTA with specific transactions and data constraints or specific QoS requirements.

In this paper, we propose a model driven framework based on the Model Driven Engineering (MDE) approach [5] for modeling new QMS by reusing models of well-known solutions. The framework focuses on a strategic methodology to help users modeling QMS by and for reuse.

The rest of the paper is organized as follows. In Section 2, we present a state of the art on modeling works related to RT DBMS. Section 3 gives a brief presentation of the QoS management architecture in RTDBMS. The QMS components are the subject of Section 4. The framework architecture is detailed in Section 5. In Section 6, we present the proposed methodology. Section 7 presents the framework implementing. In Section 8, we give the results of experiments on a concrete QMS. We end this paper with a conclusion and give some future works.

 ISBN: 978-1-61208-438-1

II. RELATED WORK

QMSs mentioned above, that we have deeply studied, are not modeled. They are described in a natural language and presented through an architecture, QoS parameters, system variables and algorithms acting to meet the QoS specification. In the RTDMS domain, there are some object-oriented models focusing on Real-Time object-oriented databases modeling, and especially on system aspects, such as RTSORAC [6], RODAIN [21] and BeeHive [20].

These works propose object-oriented models that only consider structural and behavioral features of Real-Time data and transactions. Some of them partially support the modeling of scheduling, concurrency control, Real-Time data distribution and quality of service policies. They provide only specific data and transactions models and cannot support new data and transactions constraints.

Other works offer UML profiles for modeling RTDBMS, such as RTO-RTDB [15], RTO [9][13]. They provide UML extensions for Real-Time databases designers, through stereotypes, which express both the Real-Time data and transactions constraints.

The related work that we have presented provides a rich foundation upon which we can build. However, the QoS modeling is partially supported in these models, which provide specific QoS parameters and cannot specify new parameters. They are all focused on data management and do not consider transactions management. Moreover, there is no one work that focuses on QMS modeling and reuse.

III. QOS MANAGEMENT ARCHIRECTURE IN REAL-TIME DATABASE MANAGEMENT SYSTEMS

The QoS is specified by the DBA through *QoS parameters*, which are metrics defining the desired performance of the RTDBMS. The DBA specifies Quality of Data (QoD) and Quality of Transactions (QoT) parameters with reference values they must not exceed. In transient overloads, an overshoot of these thresholds may be tolerated by giving the worst-case system performance. Here, we give an example of QoS parameters proposed in [11].

- *Deadline Miss Ratio (MR)* is the percentage of user transactions that missed their deadlines regarding to the accepted ones. MR is considered as a QoT parameter.
- *Perceived freshness (PF)* is the ratio of fresh data to the entire temporal data in a database. Fresh data are data updated within their validity interval. It is considered as a QoD parameter.
- *Maximum Overshoot (Mp)* defines the worst-case system performance in the transient system state, e.g., the highest MR in the transient state. QoS parameters can overshoot their reference values at maximum of Mp.
- *Settling time (ts)* is the time for the transient overshoot to decay and reach the steady state performance.

In [11], authors propose the following QoS specification: $MR \leq 5\%$, $PF \geq 98\%$, $Mp \leq 30\%$ and $ts \leq 45$sec.

The well-known QMS for RTDBMS use the FCSA because it provides the QoS specification guarantee without a priori knowledge of transactions workloads using feedback control. FCSA has shown to be very effective for a large class of systems that exhibit unpredictable workload. For instance, it was applied to improve the QoS in distributed multimedia systems [22] and recently in geographic information Systems [23].

The basic FCSA applied by most QMSs in RTDMS (Figure 1) was proposed in [2]. This architecture is based on the principle of observation and self-adaptation. The observation is the periodic measurement of QoS parameters by a Monitor. The auto-adaptation is the dynamic adjustment

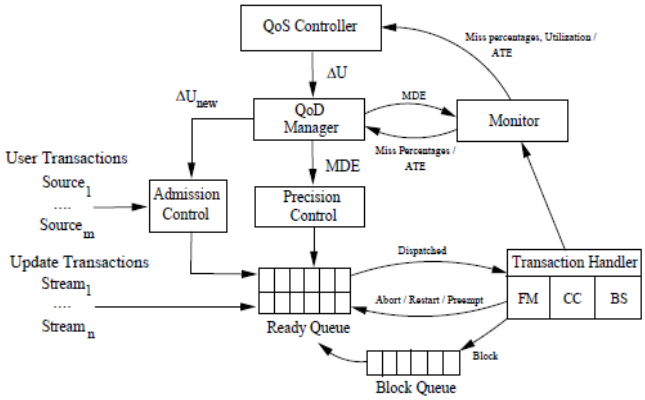

Figure 1. Basic Feedback Control Scheduling Architecture in RTDMS.

of the measured QoS to converge to the required QoS achieved by the QoS Manager. The adjustment is computed by a QoS controller containing feedback loops. The QoS Manager executes a QoS management algorithm, which implements adjustment scenarios based on system states [2].

The well-known QMSs in the literature [2][3][11][12] are efficient to guarantee the QoS specification even in transient overloads thanks to the FCSA and Imprecise Computation techniques, which allow the use of approximate results and imprecise data. However, these solutions may be inadequate for RTA requiring specific QoS requirements and using specific data and transactions models. It is more common to take a subset of their elements in order to reuse them for a new QMS design, which is not provided by related works.

IV. QOS MANAGEMENT SOLUTIONS COMPONENTS

Through the evaluation of the most referenced QMS [1], we conclude that their differences and similarities are focused on three components: *Models, Parameters* and *Policies*, which are interdependent. In addition, RTA requirements can be regrouped according to these components, which are worth of reuse to fit new requirements. For instance, take a RTA that requires a deadline miss ratio MR≤10% and its user transactions can follows the Milestone model that tolerates transaction

 ISBN: 978-1-61208-438-1

imprecision, but does not tolerate data imprecision for critical data. In this case, the suitable QMS must combine transactions and queues models from the FCS-IC1 QMS [2] with the data model, from the QMF1 QMS [11] and combine QoD parameters from QMF1 with QoT and loops parameters from FCS-IC1. Policies of QMF are suitable for this RTA with some reconfiguration. If the same RTA tolerates data imprecision, but requires specific QoS management scenarios, then the FCS-IC1 can be reused in this case, but its QoS management policy must be rewritten.

The component *Models* gathers the different models used by a solution, namely: the Real-Time transactions model, the Real-Time data model and the queues model, which vary from a solution to another. We employ the notion of data model to define data attributes, data types and data imprecision implementation. For instance, authors in [2] allow data to deviate, to a certain degree, from their corresponding values in the external environment. The transactions model denotes transactions attributes, transactions types and the transactions imprecision implementation, e.g., multiple versions, use of sieve functions and the Milestone approach [10]. The queues model describes the queues configuration through the number of queues, priority levels and types of transactions in each queue.

The component *Parameters* includes the QoD and QoT parameters, system parameters such as *ts* and *Mp* and feedback loops parameters that differ from one QMS to another. The *Parameters* component configuration depends on that of *Models* component.

The component *Policies* is based on the two previous components. It comprises algorithms which impact on the QoS, namely: the scheduling algorithm, the concurrency control algorithm, the updating algorithm and the QoS management algorithm. This later tries to balance the RTDBMS workload between user and update transactions through a compromise between QoT and QoD. For instance, if the system downgrades the QoD, many update transactions will be rejected and vice versa.

V. FRAMEWORK ARCHITECTURE

Our proposal is guided by the need to reuse some elements of QMSs in order to better meet new QoS requirement and specific constraints on Real-Time data and transactions. For this purpose, we provide to users a theoretical and practical framework allowing them design and extend their own QMS models. This framework is based on the reuse of the well-known QMS models, which are modeled by an expert using a specific editor and broken down into reusable fragments, that represents QMS components (Section IV).

A. Meta-modeling architecture

The framework provides "productive" QMS models that can be processed and transformed automatically in order to be reused. This is allowed using the OMG meta-modeling architecture [24] and models transformations [5], which are the core of the MDE approach that we adopted.

The proposed four-layered modeling architecture is based on the OMG meta-modeling architecture for the MDE (Figure 2). In the lowest layer M0 of systems, we find QMS. The layer M1 above M0 contains the models that represent QMSs, which we call QoS Management Approaches (QMA). A model in M1 must be conformant to its meta-model in the layer above, named M2, which defines all the concepts of a model and is considered as an abstract syntax of a specific modeling language to formalize model description. In this work, we designed a QMS meta-model named MM-SGQdS to formally express their corresponding QMA in a modular way that represents all QMS components. Thus, QMAs will be conformant to MM-SGQdS and can be easily manipulated and reused. To implement MM-SGQdS, we chose the EMF core (Ecore) meta-meta-model at the layer M3 (above M2) [4]. It is a subset of the OMG standard meta-modeling language MOF [17] that formalizes the description of meta-models in M2.

Let us now present the MM-SGQdS meta-model and its concepts.

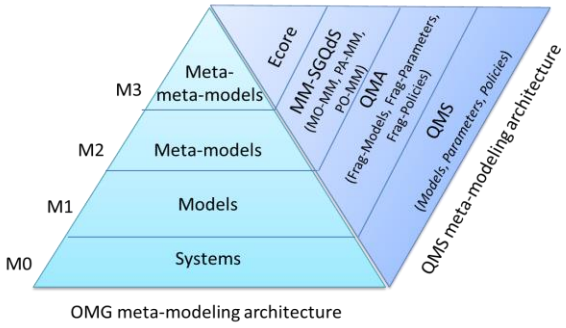

Figure 2. QoS Management Solutions Meta-modeling architecture.

B. QoS Management Solutions Meta-model

For each QMS component, namely: *Models*, *Parameters* and *Policies*, we proposed a specific meta-model (in layer M2) to formally express its elements and relationships between them. They are respectively called *MO-MM, PA-MM* and *PO-MM* (Figure 2). Thus, MM-SGQdS is comprised of these three meta-models as illustrated by Figure 3 showing its abstract view simplified.

MO-MM comprises three main meta-classes, namely: *TransactionsModel, DataModel* and *QueuesModel* describing elements of the *Models* component. PA-MM includes four main meta-classes, namely: *QoDParameters, QoTarameters, SystemParameters* and *LoopsParameters*

modeling elements of the *Parameters* component. PO-MM has four main meta-classes, namely:*QoSPolicy, SchedulingPolicy, UpdatingPolicy* and *ConcurrencyPolicy* correponding to elements of the *Policies* component. Each meta-model has a tree stucture with a root meta-class linking its main meta-classes. The root meta-classes of *MO-MM, PA-MM* and *PO-MM* are linked to a meta-class called *QoSApproach*, the root of the tree structure of MM-SGQdS.

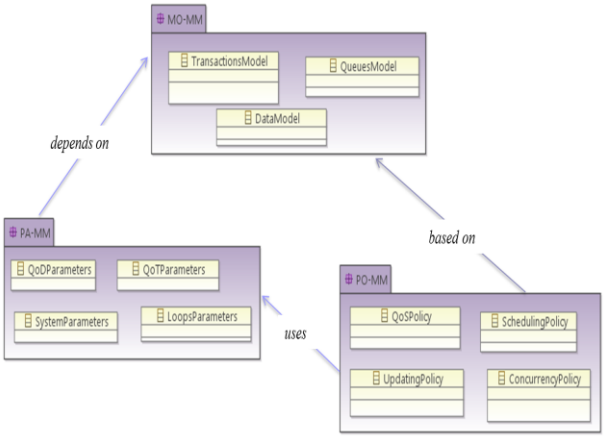

Figure 3. Abstract view of the meta-model MM-SGQdS.

A QMA conformant to MM-SGQdS is comprised of three reusable fragments called *Approach Fragments* (AF), namely: *Frag-Models, Frag-Parameters* and *Frag-Policies* (Figure 2), which represent respectively *Models*, *Parameters* and *Policies* component and conformant to *MO-MM, PA-MM and PO-MM* meta-model, respectively.

C. Logical architecture

In the software reuse domain, two kinds of actors are involved, namely: developers for reuse and developers by reuse. In this work, which focuses on QMA reuse, the framework involves two kinds of actors, namely the *RTDBMS experts* and the *QMA designers*. The *expert* generates and manages AF for reuse, while the *designer* reuses these fragments in order to build a new QMA. The two actors are guided by a methodology, which will be described later in this paper. Existing QMSs are modeled by the expert through a specific QMA editor to generate corresponding QMAs and then broken down into AF. The generated fragments are stored in a database to be reused by the designer or the expert in new situations (Figure 4).

For the decomposition of QMA, the expert uses a transformation called *Decomposition*, which generates the three AF. To build of a new QMA, the designer uses a transformation called *Composition*, which assemble AF from different QMA. The transformation called *Extension* is used to extend existing QMA by adding some elements from other approaches.

The new QMA are in turn broken down by the expert into AF using the *Decomposition* transformation.

Generated AFs are inserted into the database to be reused.

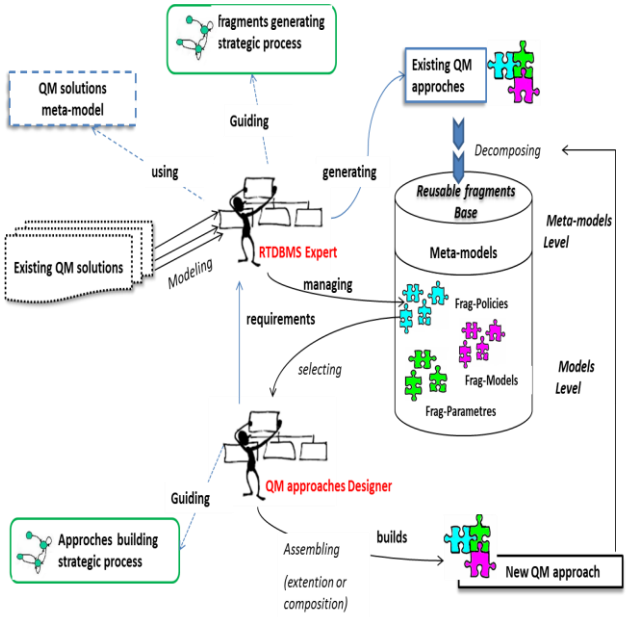

Figure 4. The framework logical architecture.

The two Processes in Figure 4 are part of a strategic methodology for QMA design, which is the subject of the following Section.

VI. Strategic Methodology for QoS Management Approaches Design

To guide framework actors to achieve their goals, we propose a methodology that defines two strategic processes, namely: GEN-PM, a process for generating AF (expert side) and CONS-PM, a process for building new QMA by reuse (designer side). These processes are modeled through the MAP formalism [19] for its flexibility and simplicity. A MAP-based process model, called *map*, is an oriented graph that represents explicitly goals to be achieved on nodes with the different strategies for achieving them on oriented labelled arcs. Subsequently, we detail the two processes models.

A. Approach fragments generating process

The map for modeling the AF generating process, named GEN-PM, contains different strategies related to three main goals, namely, *generating AF, managing AF* and *editing QMA* (Figure 5).

- *Generating AF*: three strategies are defined to achieve this goal. The strategy "*by decomposition*" aims to break down a QMA through a *Decomposition* transformation for QMA in order to extract the three AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*). Thereafter, generated AFs are stored in the database with their elements, which are generated through a *Decomposition* transformation for AF.

The strategy "*by extension*" aims to extend an existing fragment by adding it some elements with conformance to its meta-model. The expert can combine elements from different AF to build new AF, which constitutes the aim of the strategy "*by composition*". He can design a new AF "*from scratch*" for specific needs, using a specific AF editor provided by the framework.

- *Managing AF*: to achieve this intention, we set three strategies, namely: "*editing*", updates the values of an AF properties, "*deleting*", deletes un-usefull AF from the database, and "*storing*", adds a new AF in the database.
- *Editing QMA*: existing QMS are modeled through a specific editor based on their meta-model MM-SGQdS following the strategy "*by editing*".

The "*Start*" and "*End*" goals define the start and the end of the process, respectively. At the end of the process, the strategy "*validation*" is used to check the composition. Each new fragment must be stored in the database to be persistent and reused thereafter, which is the role of the strategy "*storage*".

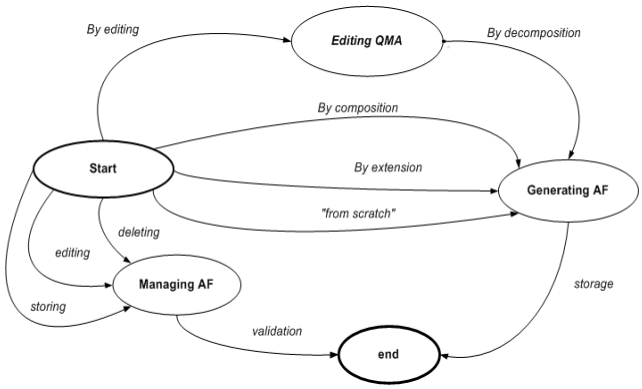

Figure 5. Map of approach fragments generating process.

B. QoS Management Approach building process

The process focusing on QMA building, called CONS-PM, is based on model reuse through model transformations. It represents designer goals and strategies to achieve them. The CONS-PM map comprises three major goals, namely *selecting AF*, *composing QMA* and *extending QMA*.

- *Selecting AF*: concerns the ways to search for AF from the database before composing them (by name, by type or by approach name).
- *Composing QMA*: builds a new QMA by linking AFs from stored QMAs or new AFs, with conformance to MM-SGQdS, using a QMA transformation called *Composition*.
- *Extending QMA*: to extend a QMA, the designer must search this approach by its name, and then he inerts into the QMA some AF elements (e.g., data model, QoD parameter, QoSPolicy, etc.), which are generated after AF decomposition, to meet RTA requirements. If necessary, new elements can be edited by the expert to be used by the designer.

VII. IMPLEMENTATION

For the framework implementation, we used the Eclipse Meta-modeling Framework (EMF) [4] to benefit from a set of tools, such as reflexive editors or XML serialization of models.

The different meta-models are implemented with *Ecore*, the EMF meta-modeling language. The model transformations are implemented with KerMeta [7]. It is an object oriented meta-modeling language. It does not focus only on structural specifications but supports the specification of the operational semantics of meta-models. Thus, models can be simulated. In addition, it is a powerful model transformation language.

A. Meta-models implementation

MM-SGQdS and the AF meta-models have a tree structure to simplify their reuse through *Decomposition*, *Composition* and *Extension* transformations. An excerpt of MM-SGQdS, displayed with the EMF reflective editor, is illustrated in Figure 6.

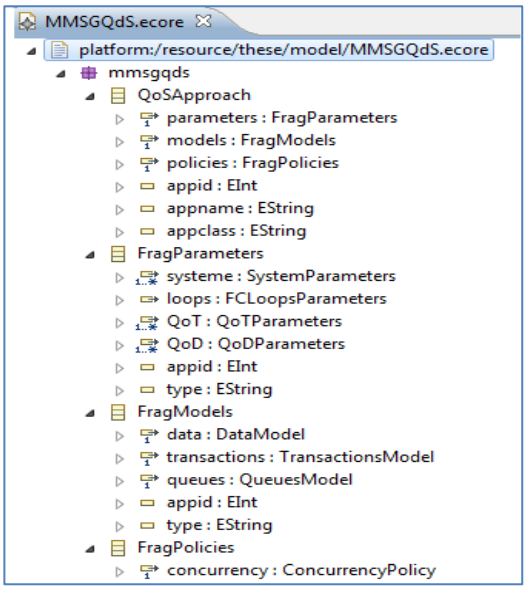

Figure 6. Excerpt of MM-SGQdS tree view.

The meta-class *QoSApproach* is the root of MM-SGQdS. The composition links *parameters*, *models* and *policies* reference respectively the meta-classes *FragParameters*, *FragModels* and *FragPolicies*, the roots of respectively, PA-MM, Mo-MM and PO-MM. The meta-attributes like *appid* and *appname* are QMA properties.

B. Model transformations implementation

A transformation is the automatic generation of a target model from a source model, according to a transformation definition. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. A transformation rule is a description of

how one or more constructs in the source language can be transformed into one or more constructs in the target language [25].

In this work, model transformations are applicable to multiple source models and/or multiple target models. We focus on horizontal and exogenous transformations called *Migration* that keep the same level of abstraction (M1) with models expressed using different meta-models [26].

Three categories of model transformations are proposed, namely: *Decomposition*, *Composition* and *Extension*. These transformations are implemented for three model levels: QMA, AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*), and QoS management policies (within the fragment *Frag-Policies*). This results in fifteen KerMeta modules. All these transformations are tested on the FCS-IC1 solution but we cannot present all results in this short paper, neither their implementations.

VIII. EXPERIMENTS

The experiments were carried out on a QMA representing a concrete QMS, called FCS-IC1 [2], which was modeled using a specific QMA editor. We tested the QMA decomposition transformation. The generated AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*) have been used to compose a new QMA, which is identical to the original QMA. We also tested the QMA extension by adding a new QoS Management scenario to the QoS Management Policy within the Frag-Policies fragment of the original QMA.

Other transformations are tested to reuse AF elements, which are very useful for designers. For instance, the *Frag-Models* fragment of the FCS-IC1approach was decomposed into three elements modeling respectively the transaction model, the data model and the queues model. These elements are recomposed to generate a new Frag-Models fragment identically to the original fragment.

A. Expert side experiments

1) Approach edition

The EMF easily generates model editor for an Ecore meta-model to create instances which conform. This allowed us generate QMA editor and editors for each AF. For instance, the QMA editor provides the expert, through a *QoSApproach Editor* menu (Figure 7), a way to instantiate the QMA root meta-class (instance of the *QoSApproach* meta-class) and to configure its related AFs conformant to MM-SGQdS.

The Edited models for QMA or AF modeling are serialized in XMI for their persistence in the database. To display models, EMF provides a graphical mode using its reflexive editor. We developed methods to load AFs from their XMI files for reuse and to display them in a textual mode, which is not provided by EMF.

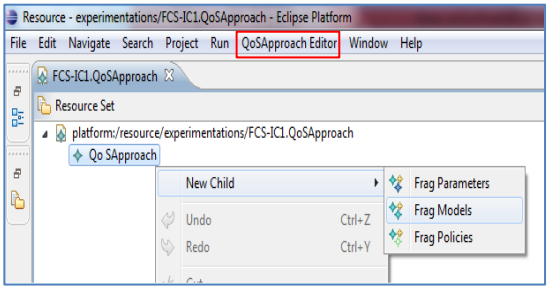

Figure 7. QoS Management Approach Editor.

2) Approach decomposition

To decompose a QMA into AF, we developed a KerMeta module named *decomposeQoSapproaches.kmt*. The method *decompose* of this module parses the QMA from its root.

For each fragment identified through a composition link (instance of *parameters*, *models* or *policies* links in MM-SGQdS) the method instantiates a new fragment conformant to its meta-model (MO-MM, PA-MM or PA-MM). Each AF is serialized in XMI for its persistence.

An excerpt of the graphical display of the Frag-Models fragment, resulting from the FCS-IC1 approach decomposition, is illustrated in Figure 8. This fragment represents the model of the Models component of the FCS-IC1 solution.

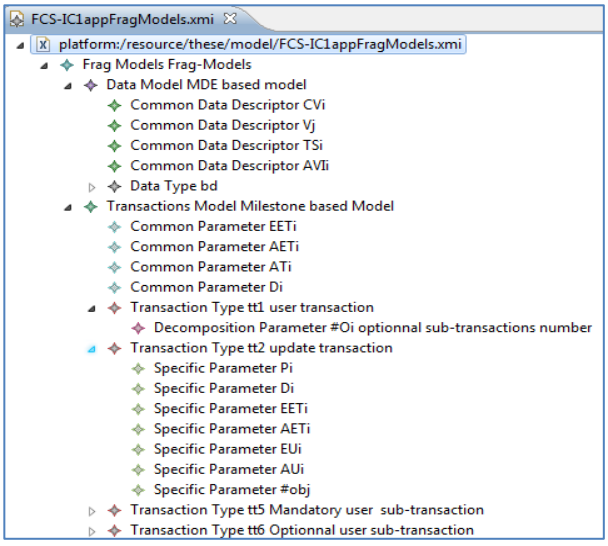

Figure 8. Excerpt of the Frag-Models fragment of FCS-IC1 QMA.

It begins with the configuration of the Frag-Models fragment with conformance to MO-MM. There are detailed the data model elements followed by the transactions model elements.

 ISBN: 978-1-61208-438-1

B. Designer side experiments

1) Approach composition

This transformation generates a new QMA conformant to MM-SGQdS. The new approach is serialized to XMI and displayed in a textual mode. The KerMeta module developed for this purpose is called *composeQoSApproaches.kmt*. It contains a method named *compose* that has a unique parameter, a string array, containing the names of XMI files of serializes AF that will be assembled. For each file, we get the tree structure of the AF by loading its XMI file and meta-model. Then, AF elements are matched to QMA elements and attached to the root of the new QMA. An excerpt of the textual display of the composed FCS-IC1 approach is illustrated in Figure 9. It has the same fragments of the original FCS-IC1 approach.

```
*** Approach : FCS-IC1-composed ***
*** fragment : Frag-Models ***
*Data model: MDE based model
        Principle: Base data are used with imprecision defined by MDE (Maximum Data Error)
        Updating policy : MDE-UpdatePolicy
        Common Descriptors:
                CVi, current value of data di
                Vj, updated value by update transcation j
                TSi, TimeStamp, latest update time of data di
                AVIi, Absolute Validity Interval of data i (AVIi=2*Pj)
        Data Types:
                bd, base data with periodic update
                Specific Descriptors:
                        DEi, data error of data di, [0%, 100%], 100*|CVi-Vj| / |CVi | %
                        AVi, Average value of data di, U(0,100)
                        Vi, value of data di, N : (AVi, AVix×VarFactor), periodic mesure
                        varFactor, data model parameter, U :(0,1), null
*Transaction model: Milestone based Model
        Principle: user transactions are decomposed into one mandatory sub-transaction and many optionnal sub-transaction
        Common Prameters:
                EETi, Estimated Exexution Time of transaction ti
                AETi, Average Exexution Time of transaction ti
```

Figure 9. Textual display of composed FCS-IC1.

2) Approach extension

We tested the extension of the *QoS management policy* of the FCS-IC1 approach (included in its *Frag-Policies* fragment) with a new QoS management scenario. For this purpose, we propose to apply on-demand updates to non-critical data when the system is at maximum overload. In this way, we can avoid the system saturation, during which all coming user transactions are rejected. This new scenario was modeled with corresponding editor (Figure 10) and added to a copy of FCS-IC1approach with conformance to MM-SGQdS.

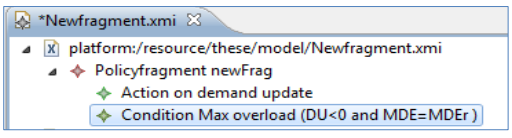

Figure 10. Model of the QoS management scenario used to extend FCS-IC1.

IX. CONCLUSION

In this paper, we proposed an MDE-based framework to automate the reuse of QMA in RTDBMS, which is useful to meet new RTA constraints and QoS requirements. The MDE aims to increase productivity by offering modeling languages tailored to a specific domain, which are typically defined by meta-models. The QMA reuse is based on model transformations, which is the core concept of MDE. Three categories of model transformations are proposed, namely: *Decomposition*, *Composition* and *Extension*. These transformations are implemented for three model levels: QMA, AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*), and QoS management policies (within the fragment *Frag-Policies*). These transformations will be detailed in an extended version of this paper. The framework provides also a methodology consisting of two strategic processes GEN-PM and CONS-PM to help experts and designers achieve their goals with multiple strategies to achieve them, depending on their requirements.

Based on meta-models, we generated editors for QMA and their reusable fragments. We implemented and tested transformations on a concrete QMA. After decomposition, no specificity is ruled out and the rebuilding leads to the initial approach, conformant to the meta-model MM-SGQdS.

Actually, we are interested in providing users a tool for querying the models in order to extract useful knowledge for reuse.

REFERENCES

[1] S. M'barek, L. Baccouche, and H. Ben Ghezala, "An evaluation of QoS management approaches in Real-Time databases". In proc. of the Third International IEEE Conference on Systems. Cancun (ICONS'08), Mexico, 2008, pp. 41-46.

[2] M. Amirijoo, J. Hansson, and S. H. Son, "Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations", IEEE Transactions on Computers, vol. 55, no. 3, 2006, pp. 304-319.

[3] E. Bouazizi, B. Sadeg, and C. Duvallet, "Improvement of QoD and QoS in RTDBs", In proc. of 14th int. conf. on Real-Time and Network System, Poitiers, France, May 30-31, 2006, pp. 87-95.

[4] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework, second edition (Eclipse), Addison-Wesley Longman, Amsterdam, 2009.

[5] J. M. Favre, "Towards a Basic Theory to Model Driven Engineering". In Proc. third Workshop in Software Model Engineering (WISME 2004), Lisbon, portugal, 2004, pp. 262-271.

[6] L. C. DiPippo and L. Ma, "A UML package for specifying Real-Time objects", Computer Standards and Interfaces, vol. 22, no. 5, 2000, pp. 307-321.

[7] Z. Drey, C. Faucher, F. Fleurey, and D. Vojtisek, KerMeta language reference manual, 2009.

[8] K. Ramamritham, S. H. Son and L. C. Dipippo, "Real-Time Databases and Data Services", Real-Time Systems, vol. 28, no. 2-3, 2004, pp. 179-215.

[9] Z. Ellouze, N. Louati and R. Bouaziz, "The RTO-RTDB Real-Time Data Model". In Proc. of The 2012 World Congress in

 ISBN: 978-1-61208-438-1

Computer Science, Computer Engineering, and Applied Computing, 2012, pp. 333-339.

[10] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise computations", Proceedings of the IEEE, vol. 82, no. 1, 1994, pp. 83-94.

[11] K. D. Kang, S. H. Son and J. A. Stankovic, "Managing Deadline Miss Ratio and Sensor Data Freshness in Real-Time Databases". IEEE Trans. Knowl. Data Eng. vol. 16, no. 10, 2004, pp. 1200-1216..

[12] B. Alaya, C. Duvallet and B. Sadeg, "A new approach to manage QoS in Distributed Multimedia Systems", (IJCSIS) International Journal of Computer Science and Information Security, vol. 2, no. 1, 2009, pp. 1-10.

[13] N. Idoudi , N. Louati, C. Duvallet, R. Bouazizi, B. Sadeg and F. Gargouri, "A Framework to Model Real-Time Databases". International Journal of Computing and Information Sciences (IJCIS), vol. 7, no. 1, 2010, pp. 1–11.

[14] K. Liu, V. C. S Lee., J. K-Y Ng, J. Chen and S. H. Son, "Temporal Data Dissemination in Vehicular Cyber–Physical Systems", IEEE Transactions on intelligent transportation systems, vol. 15, no. 6, 2014, pp. 2419-2431.

[15] N. Louati, C. Duvallet, R. Bouaziz, and B. Sadeg, "RTO-RTDB: A Real-Time object-oriented database model", In proc. of the 23rd IASTED International Conference on Parallel and Distributed Computing and Systems, Dallas, USA, 2011, pp. 1-9.

[16] C. Lu, J. A. Stankovic, G. Tao and S.H. Son, "Feedback control Real-Time scheduling: Framework, modeling and algorithms", In Journal of Real-Time Systems, vol. 23, no. 1, 2002, pp.85-126.

[17] OMG. Meta Object Facility (MOF) specification - version 1.4, formal/01-11-02, Avril 2002.

[18] K. Ramamritham, S. H. Son and L. C. Dipippo, "Real-Time Databases and Data Services", Real-Time Systems, vol. 28, no 2-3, 2004, pp. 179-215.

[19] R. Deneckere, E. Kornyshova and C. Rolland, "Enhancing the Guidance of the Intentional Model MAP: Graph Theory Application", IEEE 3rd int. conf. on Research Challenges in Information Science RCIS'09, Fes, Morocco, 2009, pp. 13-22.

[20] J. A. Stankovic and S. H. Son, "Architecture and Object Model for Distributed Object-Oriented Real-Time Databases", In Proc. 1st Int. Symp. on Object Oriented Real-Time Distributed Computing, Kyoto, Japan, 1998, pp. 414–424..

[21] T. Niklander and K. Raatikainen, "RODAIN: A Highly Available Real-Time MainMemory Database System". In Proc. of the IEEE International Computer Performance and Dependability Symposium, Durham, NC, 1998, pp. 271-.

[22] B. Alaya, C. Duvallet and B. Sadeg, "Feedback architecture for multimedia systems". In proc. of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2010), 2010, pp. 16-19.

[23] S. Hamdi, E. Bouazizi and S. Faiz, "A new QoS Management Approach in Real-Time GIS with heterogeneous Real-Time geospatial data using a feedback control scheduling". In Proc. 19th International Database Engineering & Applications Symposium (IDEAS'15), 2015, pp.174-179

[24] C. Atkinson, and T. Kühne, "Model-driven development: A metamodeling foundation". IEEE Software, vol. 20, no. 5, 2003, pp. 36–41.

[25] A. Kleppe, J. Warmer and W. Bast., MDA Explained, The Model-Driven Architecture: Practice and Promise. Addison Wesley. 2003.

[26] T. Mens, P. Van Gorp, "A taxonomy of model transformation". Electronic Notes in Theoretical Computer Science 152, 2006, pp. 125–142.

 ISBN: 978-1-61208-438-1