Easy Modeling and Fast Exploration of Multimedia Heterogeneous Applications with UML/MARTE

Authors omitted for blind review

Authors omitted for blind review

***Abstract*—Automatic generation of executable models from high-level system specifications has demonstrated to be a quite helpful tool for embedded system designers. Productivity boosts with the use of standard specification languages and design methodologies that settle common foundations for requirements capture, system definition, exploration of design space and early verification (functional and not functional parameters). In this context, this work presents a specification methodology and design flow for multimedia embedded system using UML/MARTE standards.**

First, system modeling is performed through a technology independent MARTE profile that captures application functionality. Then, the corresponding SystemC executable model is generated after a mapping transformation to a HW/SW platform description that includes reconfigurable hardware, software and bus resources.

Keywords—UML; MARTE; multimedia embedded system design; SystemC; HW/SW codesign; FPGA

I. INTRODUCTION

Embedded system design is an increasingly challenging task since their complexity seems not to reach the top ever. Integration of more functionality and applications in truly reduced time budgets, is the current way for companies to get success with their products; being the first in the market with the last technology.

Lately, this trend is even more intensive in the multimedia embedded platforms ecosystem with a cannibal scenario that includes consumer electronic devices such as smartphones, tablets, smartTVs, set-up-boxes, etc. In this case, the new systems must work under strict performance constraints, which add extra development work to verify non-functional requirements. Also, the use of hardware accelerators in this kind of platforms is becoming a must to cope with the real time requirements of multimedia application.

Therefore, the scenario depicted above is hard to be managed properly with conventional design methodologies. Heterogeneity, HW/SW integration, complexity, verification and short time-to-market windows is a dangerous, unstable cocktail.

In this work, we focus on the description of a design methodology that enables an easy specification of a multimedia system and a fast translation towards a real product or prototype. To this end, we make it use of an ESL (Electronic System-Level) design strategy [1] and MDD (Model-Driven Development) techniques [2]. Our proposal grounds in the use of standard specification languages such as UML [3] and MARTE [4] which is also a standard profile conceived to model and analyze embedded and real-time systems.

Following the philosophy established by MDD, a platform-independent model must be specified in first place, which must capture the functionality of the system. A UML profile for multimedia heterogeneous platforms specification has been developed, reusing the design artifacts already present in the above mentioned standards. The goal was not to create a new profile but to make the most of the existent, widely used standards to ease the adoption of our solution.

The proposed profile enables the designer to completely specify the most relevant multimedia system attributes to: (a) define an application model that could be easily mapped to current standardization efforts and; (b) to obtain, in an automatic way, the SystemC [5] executable specification to explore the solution space.

Regarding to (a), it is worth mentioning that our proposal effectively integrates the major abstractions present in the *OpenMax* [6] and *GStreamer* [7] initiatives that can be summarized as: a component architecture that implements synchronous and asynchronous data exchanging mechanisms based on the producer-consumer paradigm.

Although specificity has been avoided as much as possible in the profile definition, a special consideration has been taken to the OpenMax standard. One of the reasons of this decision is the rapid adoption OpenMax is experiencing in many commercial products of leading companies: NVIDIA [8][9], Texas Instrument [10] or Adaptive Digital [11]. This is attractive to us since it is an opportunity to demonstrate the viability of this work. The aim is to extend and complete the work initiated with the implementation of a *HW OpenMax Integration Layer infrastructure* [12] providing the designer with the capability to design, explore and generate OpenMax based applications that will run on a heterogeneous platform.

Therefore, the specification of the other two platform models, required by MDD methodology (definition and specific), are also influenced by this approach.

After successive refinements, a SystemC executable specification is obtained from the platform-specific model. SystemC is used in this work since it is a modeling language that can be applied to architectural exploration, performance estimation, functional verification and HW and SW generation. SystemC model generation is accomplished, unlike many other approaches, using a correct by construction strategy instead of annotating or using specific profiles in the source UML diagram [13,14,15]. This focus keeps clean and simple the platform models.

The design space exploration is performed feeding the SystemC executable with a set of test-benches that are generated as well using another UML standard, the UML Testing Profile [16].

Paper is organized as follows. After the presentation of the motivations and objectives of this work, the proposed UML/MARTE profile is described in section II. In section III the design flow is described and section IV details the SystemC component specification that has been followed in this paper. Finally, section V provides an overview of the execution model and how the design space exploration is carried out.

II. UML/MARTE PROFILE

In this section, we are going to sketch the main modeling artifacts that have been captured in a UML/MARTE profile for multimedia embedded systems specification.

As in the standard, the information of the model is organized in separate concerns. Each concern has its correspondence with a model view with the appropriate graphical representation. MDA principles define three different points of view:

- Platform Independent Model (PIM). The PIM captures system functionality and, in this case, the component-oriented architecture of our platform.
- Platform Description Model (PDM). The PDM details the PIM view with information about: HW or SW implementation of the application components, specific data exchange mechanism, etc.
- Platform Specific Model (PSM). The PSM describes the mapping to a specific technological platform.

A. PIM: describing the application structure

The reference model adopted in our proposal envisions a multimedia system topology as a chain of *components*. Components connect with each other through *ports* that define a communication channel with specific semantics for data *buffer* exchange. These, briefly, are the main model concepts that are defined at this level. Figure 1 shows a simple example of an application PIM.

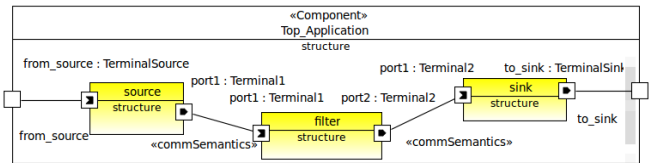

Figura 1.- Example of a application PIM.

Application components are defined using the «RtUnit» MARTE stereotype. There are three different types of components, depending on the number and direction of their communication ports: sources, filters and sinks. Ports are specified by the MARTE stereotype «FlowPort».

The behavior of each component is characterized at this level by the following characteristics: (a) latency, the time needed by the component to process a buffer; (b) input rate, the minimum time between buffer arrival; (c) data ratio, the factor input data is multiplied (i.e. decoders) or reduced (i.e. encoders); and (d) output rate, which is derived from the previous parameters. To model these attributes, two MARTE stereotypes are used: «RtSpecification» and «GaLatencyObs». There is only one feature left in this «ConcurrencyView», the size of the internal memories that the components owns to store temporary data. In this case, it is used the *memorySize* attribute of the *RtUnit* stereotype.

Just with this information, we could only be able to generate a model of the component's behavior. An application also depends on the communication mechanisms implemented to interconnect components. This is the goal of the specifications gathered in the «CommunicationView» UML package. Communication is determined by: (a) the data buffers exchanged by the components which can overwhelm the processing capabilities of the components and cause congestion in the communication channels (e.g. not enough memory capacity); and (b) the semantics of the communication, the specific mechanisms used to exchange buffers.

A port supports only one type of buffer. A buffer is modeled by means of the stereotype «StorageResource» in the MARTE standard. A buffer is defined by the type of data it holds (attribute *elementSize*) and its capacity (*result* attribute). A component port is a «CommunicationEndPoint» stereotyped UML component. Recall that the component *FlowPorts* are typed by a *CommunicationEndPoint*. Fig. 2 represents how buffer and port communication endpoints are modeled in the profile.

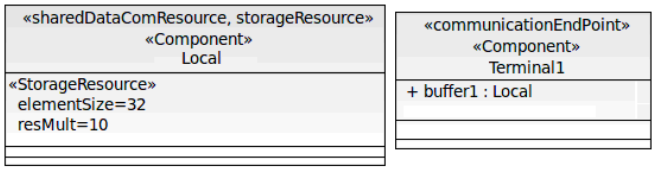

Figure 2.- Communication view definitions

Finally, to specify the communication semantics associated to the channels that bind two ports (represented as the UML connectors, see figure 1), the «CommSemantics» stereotype is

applied. Two different pairs of communication parameters are considered at this level: (a) tunneled or not; and (b) synchronous (blocking) or asynchronous (non-blocking). To this end, the «CommunicationMedia» MARTE stereotype has been extended.

B. PDM and PSM: heterogeneous platform description

At this level, new information related with the implementation details of the PIM are introduced. First, we consider the specification of the HW or SW nature of the application components.

On one hand, the «SWPlatformView» includes the UML components specified by the MARTE stereotype «SWResource». On the other hand, the «HWPlatformView» relies principally on the use of the «OpenMaxComponent» stereotype. We now focus on the attributes that define an application component that is going to be implemented in the reconfigurable logic fabric of our prototyping platform.

Resource usage and performance features are modeled using the standard «HwResource» and «ResourceUsage» MARTE stereotypes together with the UML standard stereotype «File» (see figure 3). A bitstream is a binary file that is used to program the FPGA in order to configure the logic gates to implement a hardware component. The bitstream file provides data about how much resources will be needed: LUTs, Flip-Flops and Slices.

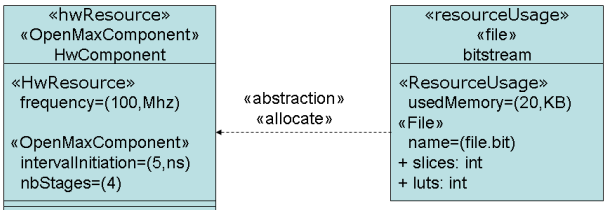

Figure 3.- Hw component modeling

The «OpenMaxComponent» defines the following attributes regarding the so far unknown internal architecture details of the hardware application component: *intervalInitiation*, *nbStages*, *baseAddress*, etc. It is worth noticing that some of the attributes defined at the PIM level still applies (for example, the compression factor). This makes the model to remain simple and reuse as much information from other stages of the specification methodology.

As to the communication semantics, the *OpenMaxComponent* adds information about the size of bus transactions to transmit the data (*burstLength*) or the optimizations applied to the bus-based communication (*mode*).

Once the SW and HW resources have been defined, the *RtUnit* components are mapped onto these HW/SW resources. In the package «ArchitecturalView» a UML component is included where application instances defined in the PIM are allocated onto HW/SW resources.

By using UML abstractions specified by the MARTE stereotype «Allocate» this application-HW/SW resource mapping is modelled (figure 4).

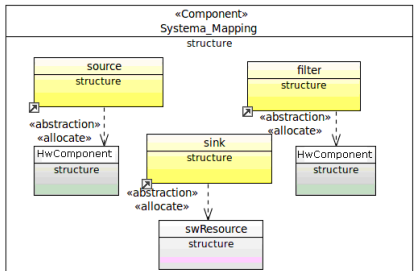

Figure 4.- HW/SW resource mapping

III. DESIGN FLOW

In the early stages of design, the behavior of the heterogeneous system and its performance are difficult to quantify. The help of a system model can be very useful in order to evaluate the different design alternatives, reducing the development costs. SystemC is the modeling language that has been used to develop the system model in this work. The strength of SystemC is its applicability to almost every stage in the design and implementation flow: system-level specification, architectural exploration, performance modeling, software development, functional verification, and high-level synthesis,

Starting from the UML/MARTE model of the component-based system, the proposed design (figure 5) flow generates all the SystemC code required for the simulation and evaluation of the system. Papyrus [17] is a graphical modelling open source UML2 tool based on Eclipse environment that has been used to create the UML/MARTE model. A code generator has been developed as a set of generation templates written in the standard MTL language [18]. The development has been done through Acceleo [19], a code generation framework fully integrated in Eclipse.

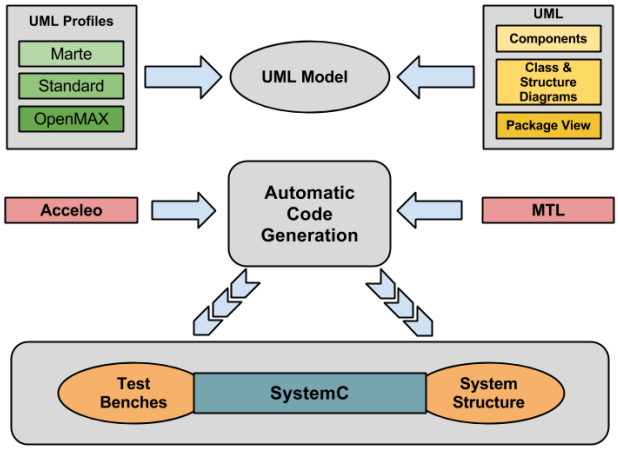

Figure 5.- Proposed OpenMAX Flow

The main goal is to know, through the automatic generation of the SystemC specification, the temporal

behavior of the system, keeping the different design alternatives in mind. In order to select the most adequate alternative for the components configurations, this SystemC specification is simulated in the proposed simulation platform.

The characteristics of each component in the system are registered in the model UML/MARTE, as it has been explained in the previous section. The code generation process produces a SystemC specification for each component (HW or SW). Next, we provide the reader with a more detailed explanation of the application component modeling using SystemC.

IV. SystemC Component Specification

Each component (figure 6) in the system integrates two main parts. In one hand, the implementation of the multimedia function called Media Core (MC), and in the other hand a placeholder for this MC called System Adapter (SA).

Internally, the MC has a parametrized multi-stage pipeline making the temporal behaviour of the MC flexible and configurable since the number of stages and the cycle time can be set. This pipeline reads media data in its inputs and generates new data in its outputs. The MC should be independent of the data communication mechanisms and the memories technologies in order to improve its reuse opportunities. Such independence is provided by the SA that contains two local memories, from which the MC reads and writes buffers (minimum amount of media data exchange between two components), and provides the connectivity to the rest of the system. A simple fixed interface is provided between the MC and the SA, mainly based on a memory interface.

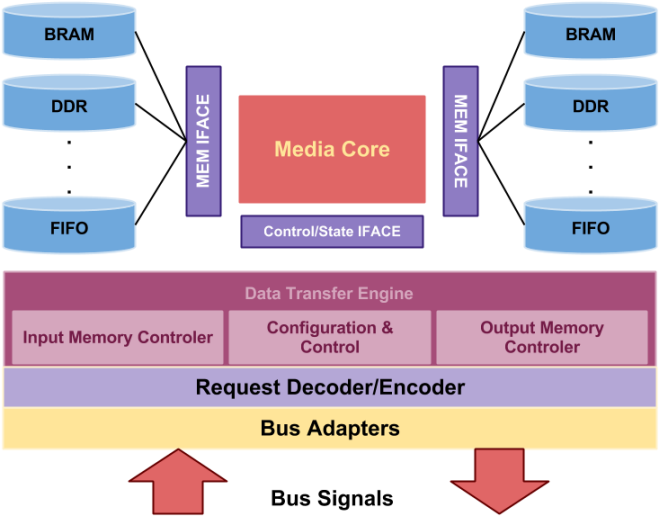

Figure 6.- Component Model

Besides the memory interfaces, the SA implements the control logic that governs the MC execution (start or stop signals between others). Also, the MC provides information to the SA about its execution status, for example read_done signal indicates that an input buffer has been consumed (not necessary processed) and write_done tells the SA that the MC has written the last word of the output buffer. Both signals are interpreted by the Data Transmission Engine (DTE) to overlap memory operations with the data transmission process. The DTE behavior is customized with the PIM attributes described in the *CommunicationView*.

The main idea of the data transmission process is to transfer as soon as possible a buffer content between two components, that are connected through a shared bus, without the intervention of Sw routines. The communication meets the producer/consumer pattern and the components involved in the communication exchange "FillBuffer" and "EmptyBuffer" message types. These messages are translated into bus requests by the Bus Adapters.

A SystemC component template (figure [7]) based on the above model has been developed. This component template is flexible and allows a high degree of configurability to obtain a temporal behavior similar to any candidate component that can be instantiated in a real system. This helps the designer to get a close performance profile of the target system.

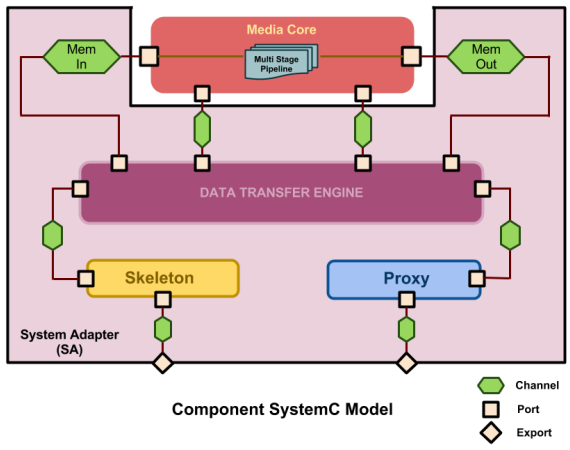

Figure 7.- SystemC OpenMAX Component specification.

As it is required by SystemC, each part of the component is instantiated as a module. Communication between modules is performed through channels and ports. The channels are associated with a parameter of time to characterize the communication delay between modules.

The main difference between a SW component model and HW component model, in terms of behavior, is the temporal variation for the same task. In our concept of SW

component, this variation happens because: (a) a SW component uses the global memory system as buffer storage; (b) the processing time is higher than in HW; (c) I/O delays; and (d) the operating system stack. Therefore SystemC channels definitions support these temporal parameter variations.

Different templates for HW and SW components have been taken from a component library according to their temporal requirements. Besides, the memory access time of the local memories present in the SA and their capacities are parametrized too.

The chain composed by HW and SW components will be evaluated in the SystemC simulation platform that has been generated automatically from UML/MARTE model.

V. SystemC Simulation Platform

The simulation platform (figure 8), developed in SystemC as well, is made up of a chain of SystemC components (HW and SW) connected through a generic model bus. This generic model can be easily customized to emulate specific or commercial protocols at transaction level. For example, in our prototypes we use a configuration that targets the AHB requirements. The simulation starts by injecting media data into the first component in the chain. These data will go through components and will be collected at the end of the chain where a comparison with the golden model results is done.

During the test execution, the different SystemC modules gather statistical information that is relevant for them. This information is registered for further analysis in order to obtain meaningful conclusions about the performance of the system configuration under evaluation. The report presented to the designer includes: total application time, final throughput, average processing time per buffer and per component, average waiting times due to bus congestion, total number of transactions, overload due to synchronization messages, actual input and output buffer rates per component and average waiting time per component due to local memory saturation (not ready to received or not ready to write).

A data generator has been included in the platform that apart from providing the input data, configures and activates all the SystemC components in the system. The generator will be connected to the communication bus as well as the collector, whose mission is to receive the data that have been processed in the chain and to compare them with the expected result.

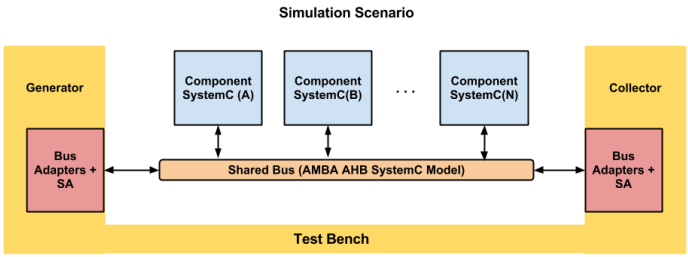

Figure 8.- Simulation Scenario

In the platform, each component is accompanied by at least two adapters (one master and one slave) that allow the connection between the component and the bus. In this way, the model of the communication bus can be replaced effortlessly, keeping the rest of the system intact. Once all the components have been instantiated, the binding to the shared bus takes place and the simulation process can start (as shown in the following source code).

```
//Sw Component parameters
struct SwParameters {
int baseAddr;
int bufferSize;
int burst_len;
int MCstages;
....
};

//Example of Hw and Sw instances
SwComp SwComp_i("SwComp_i",SwParameters);
HwComp HwComp_i("HwComp_i",HwParameters);
// Test-bench instance
testb testb_i("myTest");
//Bus Adapters declarations and bus instance
ahb_simple_master< 32 > *BusMasterAdapter;
ahb_simple_slave< 32 > *BusSlaveAdapter;
ahb_simple_bus<32> AHB_Bus = new ahb_simple_bus<32>("AHB_Bus");
//Bus binding example of Component (Hw or Sw)
//testb_i must be bind in the same way
BusMasterAdapter->master_sock(AHB_Bus->slave_sock);
BusMasterAdapter->prtProx(HwCompo_i.prox_xport);
BusSlaveAdapter->prtSkel(HwCompo_i.skel_xport);
AHB_Bus->master_sock(BusSlaveAdapter->slave_socket);

//Start Simulation
sc_start();
```

Figure 9.- Example of HW/SW Components and Bus instantiation Source Code

As it is seen in the source code of figure 9 , each component can be configured by a set of parameters that are registered in the (SW/HW) Parameters structure. The parameters that have been taken into account for the simulation and components are represented in the table 1.

Simulation Parameters	
Number of component in the system	Memory Type
Amount of input data	Communication Bus Type
For each component in the proccesing chain	
Nature (Hw or Sw)	Size of Input Memory
Base Address	Size of Output Memory
Input Ports Definition	Compression Factor of data
Output Ports Definition	Burst Length
Stages Number of the Internal Pipeline	Inicialitation Interval
Process Time per Data	
Only for Hw Components	
Local Memories Type	Data transmission mode

Table1 .- Set of simulation parameters

In order to automate the process of running different simulations with different configurations, we have been developed a factory of test cases. A XML file is used as input to the factory and it is generated from the UML/MARTE «VerificationView» which has not been covered in this article due to space constraints. All the possible combinations to be evaluated (each one is considered a test case) are captured into the XML file. Last, for each test case a simulation is launched with the intention of recording its performance and comparing it with other simulations results.

As it has been mentioned in the introduction, OpenMAX standard support is fully integrated from the beginning in the specification methodology and tools developed in this work. Therefore, it is worth mentioning the advantage that represents the simulation of component-based scenarios that are compliant with OpenMAX. Also, this platform allows the simulation of sub-chains of components over the same bus system or systems that require a bus hierarchy.

VI. CONCLUSIONS

The paper presents an UML/MARTE methodology with sufficient modeling capabilities in order to enable the design of current multimedia systems according to the specification requirements that the standard OpenMAX provides. This UML/MARTE methodology enables the system application structure to be modelled, defining all the structural and communication semantics characteristics which completely specify an application component. Then, the UML/MARTE methodology establishes a mapping to HW or SW resources of the target board. In addition, the high-level modeling methodology enables the specification of test-benches which are used to establish a design exploration process in order to find the best configuration of the system.

Then, from the UML/MARTE elements selected for this methodology, a SystemC mapping is produced. This SystemC mapping enables the automatic code generation of a SystemC executable specification in order to obtain the different timing execution performances. Depending on the values obtained, the best system configuration can be selected.

From the on-going work presented in this paper, the code generator will be implemented. This code generator will be implemented by using MTL. Then, it will be included in an Eclipse infrastructure, which will enable the application of the UML/MARTE OpenMAX methodology in the complete design of multimedia systems.

REFERENCES

[1] G. Martin, B.Bailey, A. Piziali. ESL Design and Verification: A Prescription for Electronic System Level Methodology (Systems on Silicon). March 9, 2007. ISBN-10: 0123735513.

[2] H. Kopetz. The Complexity Challenge in Embedded System Design. In 11th IEEE ISORC.

[3] Y. Vanderperren, W. Mueller, and W. Dehaene, “UML for electronic systems design: a comprehensive overview,” Design Automation for Embedded Systems, vol. 12, no. 4, 2008

[4] OMG. MARTE Profile 1.1. http://www.omgmarte.org/

[5] SystemC website. http://www.accellera.org/

[6] OpenMax Khronos. http://www.khronos.org/openmax/

[7] The Institution of Electronics and Telecommunications Engineers. IETE Technical Review. ISSN : 0256-4602. Mar-Apr 2011.

[8] NVIDIA Demonstrates High Definition Processor. Las Vegas, Nevada. January 4, 2006

[9] NVIDIA Khronos Apps SDK, 2010. http://www.nvidia.com

[10] TI software makes development easy for DM8168 and DM8148 DaVinci™ digital media processors. Technology for Innovators 2011. http://www.ti.com

[11] Adaptive Digital Technologies, Inc. Adaptive Digital OpenMAX IL Implementation. 2012. http://www.adaptativedigital.com

[12] Barba, J.; De la Fuente, D.; Rincon, F.; Lopez, J.C., "OpenMax Hardware Native Support for Efficient Multimedia Embedded Systems", IEEE International Conference on Consumer Electronics, Las Vegas (USA). Pages: 433-434. ISBN: 978-1-4244-4314-7

[13] F.Bruschi, E. Di Nitto, D. Sciuto. SystemC Code Generation from UML Models. Forum on Specification and Design Languages ’02.

[14] S. Bocchio, E. Riccobene, A. Rosti,P. Scandurra. A SoC design flow based on UML 2.0 and SystemC. In DAC, Workshop UML-Sock’05

[15] W.Muller et al. The SATURN approach to sysML-based HW/SW codesign. IEEE Annual Symposium on VLSI, ISVLSI 2010.

[16] OMG. UML Testing Profile (UTP) 1.1. http://utp.omg.org/.

[17] http://www.papyrusuml.org/

[18] OMG. MOF Model To Text Language. Jan., 2008.

[19] Website. www.acceleo.org. Nov., 2010.