A Meta–Modeling Concept for Embedded RT–Systems Design*

Benito Liccardi Thomas Maier–Komor Muslim Elkotob
Johann A. Oswald Georg Färber
Institute for Real–Time Computer Systems
Prof. Dr.–Ing. Georg Färber
Technische Universität München, Germany
{*Benito.Liccardi,Thomas.Maier–Komor,Muslim.Elkotob,Hans.Oswald,Georg.Faerber*}*@rcs.ei.tum.de*

Abstract

The work in progress presented in the present paper is focused on the HW/SW Codesign of embedded systems with a special emphasis on the Bottom–Up design process. A Meta–Model concept is introduced that gives the ability to create strict, computable and unambiguous component description, based on the OMG Standard MOF and the W3C recommendation XML Schema. The concept is currently being implemented and verified with a real world embedded system for medical applications.

1. Introduction

The domain of medium volume systems is characterized by a complex heterogeneity of application specific constraints. Methodologies that are well understood in the rapid prototyping or high volume system domain fail in this area, because of their inflexibility and scarce adaptability.

The key issues for developing these medium volume systems with a definable time to market is based upon two main aspects: *abstraction* in the design process of hardware and software and *reuse* of already designed entities with the help of architecture platforms [6, 13]. While the former is mostly controlled by a top–down design process, the latter is also driven by a bottom–up process.

The problem with the bottom–up part is mainly that it can be characterized as a *reverse engineering* task. The designer has to acquire knowledge from documents in order to be able to know if the composition of the selected system components will fulfill the requirements. This task is not only error prone but also done from the very beginning in case some major component substitution will happen. Figure 1 shows the dilemma of the reverse engineering gap during the development process.

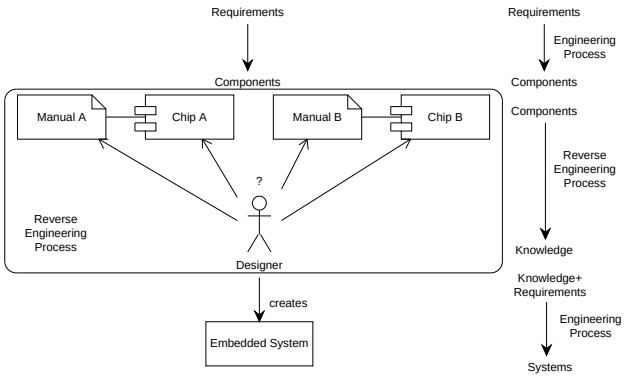

Figure 1. Reverse Engineering Gap

The designer has to cope with informal, uncomputable and different manual types, that sometimes do not include all the information that is needed and sometimes even are not up to date. Additionally complex ICs have fairly extensive manuals, which often lead to conflicts (e.g. double assignment of an I/O Pin or an unavailability of HW functions due to already used timers) and which in turn can cause expensive redesigns within very late phases of the design process.

Thus we developed a Meta–Model called ATLAS (Architecture Templates for Embedded Systems Design Meta Model)[1] for describing already existing hardware and software components on a very abstract level. This abstraction level allows us to provide strict, computable and unambiguous component descriptions. The result is that the *ability of composition* is realized by well defined semantics of the components and architectures.

*The work presented in this paper is supported by the *Deutsche Forschungsgemeinschaft* as part of a research program on "Embedded Systems" under Grant Fa 109/12-3.

[1] the Architecture Template will not be described in this paper, but is built upon the Meta–Model that is introduced in chapter 2

Furthermore architecture explorations concerning hardware compositions and software mappings are sped up.

This paper is organized in the following: The next chapter describes the ATLAS Meta–Model. Afterwards the Meta–Model realization with MOF is explained. Finally, an example for a DSP based medical embedded system is given, whose architecture is going to be explored and optimized with the ATLAS concept.

2. ATLAS Meta–Model

The ATLAS Meta–Model defines the uppermost meta structure for structural and behavioural composition. It is based upon two definitions, which are represented by two classes in the Meta–Model. This allows a separated definition of offered behaviors, required support from other devices and interactions between devices and parts.

2.1. Terminology

Definition 1 (Service) *A* Service *is defined by a functional behavior specified by a standard, a de-facto-standard, an in-house standard or by a scientifically well founded technical domain.*

Definition 2 (Entity) *An* Entity *is defined as a structural composition, e.g. an electronical part, an assembled piece of hardware, a hardware– or software–IP.*

Entities are associated with two sets of *Services*: requirements and provisions. The requirements represent the constraints on the environment, *Services* which are needed for the *Entity* in order to work properly. Provisions however, are *Services* that are implemented by the *Entity* instance itself and provide the necessary support of other devices. The provisions are the offered functionalities and behaviors that can be used by other *Entities*.

A necessary condition to find matching components is that the core unit is described in a fashion that includes all of its communication- and other peripheral-units that are available. As a result, each of those units is described as a service (see figure 2). Sometimes however, these units are only usable mutual exclusive, because they either use a pin or core-components in common. Further on it might be necessary to attach additional parts to the core-component, in order to make some unit work (e.g. Ethernet or serial controller, which requires a transceiver).

These constraints can both be modeled very easily using an association (see figure 2).

2.2. Retrieving

At the design level of a project there are usually some points that are known beforehand and therefore fixed very early in the design process. This could be the case if there is already existing experience and knowledge with a special processor core or by the need of reusing legacy code.

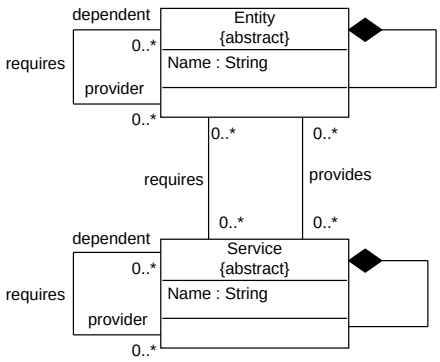

Figure 2. UML-Metamodel of the *Entity*/*Service* Relationship

These initially determined *Entities* serve as starting components for the architecture generation. The algorithm which should be applied by the designer is described by two strategies:

1. Selected components show their dependencies in form of *required Services*. Following, a search can be started to find *Entities*, that implement some or all of these *Services*. These *Entities* in turn produce additional *required Services*.

2. It is also possible to specify the system, using only *Services* instances in a first step and searching matching *Entities* in a second step.

Both strategies can be used mutually in the *iterative process* of searching suited components, which is supported by the experience of the designer. An application specific optimization (e.g. for size - using only SMD-parts; for the operation environment - using only parts suitable for military or automotive applications) is provided to keep the solution space manageable.

3. Realization with MOF

Before it is possible to find and retrieve the required components, the *Entities* and *Services* objects have to be generated according to our Meta–Model.

For this reason we applied MOF (Meta Object Facility) [12] which is a standard, released by the OMG[2]. In the MOF Meta–Model we identified similar structures and methods that we used to describe *Entities* and *Services*, as well as the methods for retrieving the components.

The first tool–support on this topic that we are going to use is dMOF[7], an OMG Meta Object Facility Implementation by DSTC[1]. It has been evaluated and it fulfills our needs for the overall framework.

3.1. Meta Layers

The four layer meta-data architecture of MOF is conveyed in the following:

L3 At the uppermost layer we have the MOF Meta–Meta–Model. A detailed description can be found in [12].

L2 The ATLAS Meta–Model (see figure 2) comprises our upper-level *Services*, *Entities* and interfaces to the system. It will be implemented mainly in MODL[2].

L1 The model–layer contains the Extensible Markup Language (XML) Schema [14, 15, 16] files that define the document structure of *Entities* and *Services*. *Services* that are defined in this layer are e.g. DSP, DMA, AD, DA, PWM, IrDA, OPAMP, SW–UART, etc.. These schemas have already been developed and are currently extended to further *Service* definitions. The main focus is to create document structures that allow the creation of documents conforming to some standard (e.g. EIA/TIA 232, CAN 2.0B /ISO11898). For *Services* which can not directly be mapped to standards or de-facto standards, we generated a document structure according to reasonable parameters that are well understood in the world of embedded systems[8].

L0 This layer contains XML data files-corresponding to entities which are the real component models. These XML files conform to the XML Schemas that have been composed and the files can be either stored in a database or in a human readable form in the file-system.

The decision for using XML and all its related applications in layer **L0** through layer **L1** is our own choice; the idea behind this was to make use of the structural and notational complexity power of XML Schema. The purpose of the overall integrated ATLAS Meta–Model in layer **L2** is to follow the Meta–Model standard of the OMG.

The current implementation of the dMOF Framework generates on the one hand IDL Interfaces and the corresponding CORBA server implementation for the metadata components, and on the other hand XMI Document Type Definitions (DTD) [11] for the specified MOF Meta–Model. The problem is, that we shifted our *Entity* and *Service* description one year ago from the less precise DTD to the more sophisticated and powerful XML Schema descriptions. As a consequence we have a break in the technology integration but, concerning the request of proposal of the OMG (see [3]) this issue is only a matter of the new version of dMOF.

[2]MODL is the language provided by the dMOF Framework

4 Application Example

For the verification of our Meta–Model we are currently examining a medical embedded system for objective estimation of human hearing loss, which has been developed at our institute in cooperation with an industrial partner. The methods implemented in this system are the *distortion product otoacoustic emissions* (DPOAE) and *brainstem evoked response audiometry* (BERA). A detailed description of the physiological background of the two methods can be found in [5]. The technical system can be described as the following:

Analog Channels[3]: DPOAEs can be evoked by two independent sinusoidal sound sources per ear in both ears simultaneously. A third sound source per ear should be used for detailed analysis of DPOAE in normal hearing subjects. Frequencies between 500 Hz and 8 kHz and a wide level range between 20 dB and 65 dB SPL must be applied to process the evoked DPOAE reasonably. BERA data is recorded by three electrodes attached to the patient's head while stimulating the ears with clicks and white noise respectively. Signal to noise ratio (SNR) is crucial to the processing of all sampled data.

Digital Channels: Filtered and processed data is transferred to a host computer by infrared communication, where the final analyses are done.

The algorithm that has been described in section 2.2 can for example be applied to one channel in the following way:

1. *Entities*, e.g. CPU or DSP cores, are instantiated beforehand, because each developer has a preferred architecture core which he wants to use. The available tool chain is also a "must" not only a "nice to have".

2. The analog signals have to be amplified, sampled and transferred to the execution unit, thus some kind of *OPAmp Service*, *DA Service* and a *BUS Service* has to be instantiated (see figure 3).

3. If some entity is found that is able to provide most of the required services, this entity can be chosen with the given constraints (area, temperature range, costs per unit, etc.) by the designer. The chosen entity will then show:

[3]A channel is defined as a data– or control link between the embedding system and a computational unit of the embedded system; in this case e.g. a microphone and the DSP or the DSP and the Infrared LED

- which kind of services it provides altogether (e.g. AC'97–link[9]) and
- which kind of services it requires (*power supply service, clock service, capacitance service, etc.*)

After the last two steps have been carried out repeatedly, this process–model leads to a structure of *Entities* and *Services* which implement the required behavior and functionality with the given non–functional constraints.

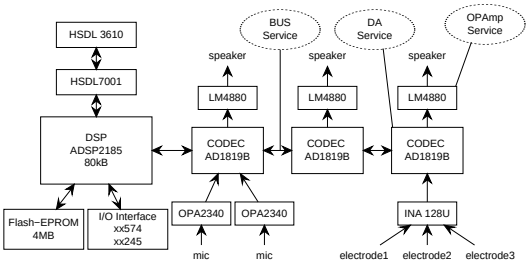

Figure 3. DSP system for DPOAE and BERA measurements

The system in its first version has not yet been explored automatically. Currently we are generating component descriptions for each *Service* and *Entity* which is used in the system.

In the existing system that we analyzed, we identified 30 different *Services*. These services provide the needed knowledge that has been used for the construction of the first prototype. The chosen components however, offer further *Services* which are not used in this version and are subject for optimizations.

The following optimization scenarios will be applied to the first prototype in the future work:

1. One can think of other DSP–Vendors or CODECS which fulfill the same *Services* as the existing system. Concerning the overall number of hardware components on the board (without capacitors, resistors etc.) we will have approximately $1.1 \cdot 10^{48}$ feasible combinations without even modifying the existing structure. An iterative approach of the algorithm described in section 2.2 will reduce the amount of combinations and therefore reduce the solution space.

2. modification of *Entities* and/or structure in order to lower the power consumption and reduce the area due to the fact, that the system should be used in a mobile headset.

3. moving the current software UART to a hardware solution due to the fact that the interrupt *Service* of the DSP is used very frequently by the communication to the CODECS and to the UART. The overall DSP utilization for the SW UART *Service* ISR comes to approximately 4 %, but the worst case response time is $2.1\mu s$ and therefore causes a bottleneck in the run–time system of the DSP software.

5 Conclusion and Future Work

The aim of the Meta–Model concept is to close the *reverse engineering gap* and to find bottlenecks in the design process very early, due to a strict, computable and unambiguous description of real components. It will also be possible to explore some architecture composition either in a generative or in a very concrete way, without having to cope with time penalties in the design process.
The concept will also have to be applicable to emerging CASE standards for embedded systems design as e.g. [10].

References

[1] Distributed Systems Technology Centre (DSTC). http://www.dstc.com.

[2] Object Management Group. http://www.omg.org.

[3] XMI Production Of XML Schema RFP. http://www.omg.org/techprocess/meetings/schedule/XMI_Prod._of_XML_Schema_RFP.html.

[4] Analog Devices. *Designers' Reference Manual*, 2001.

[5] Association for Research in Otolaryngology (ARO). *Hearing Threshold Estimation in Cochlear Hearing Loss Ears by Means of Weighted Extrapolated DPOAE I/O-Functions*, 2002. Posterpresentation 774 of the twenty-fifth annual midwinter research meeting.

[6] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution, A Guide to Platform–Based Design*. Kluwer Academic Publishers, 1999.

[7] DSTC Pty Ltd, http://www.dstc.com/Downloads/CORBA/MOF/dMOF1_1.UserGuide.pdf. *dMOF 1.1 User Guide*.

[8] P. Horowitz and W. Hill. *The Art of Electronics*. Cambridge University Press, 2nd edition, 1999.

[9] AC'97 Component Specification v2.3. http://developer.intel.com/ial/scalableplatforms/audio/.

[10] Object Management Group (OMG). *UML^{TM} Profile for Schedulability, Performance, and Time Specification*. http://www.omg.org.

[11] Object Management Group (OMG). *XML Metadata Interchange (XMI) Specification*, 1.1 edition, nov 2000.

[12] Object Management Group (OMG). *Meta Object Facility (MOF) Specification*, version 1.3.1 edition, November 2001.

[13] R. Seepold and A. Kunzmann, editors. *Reuse Techniques for VLSI Design*. Kluwer Academic Publishers, 1999.

[14] W3C. *XML-Schema Part 0: Primer*, recommendation edition, May 2001.

[15] W3C. *XML-Schema Part 1: Structures*, recommendation edition, May 2001.

[16] W3C. *XML-Schema Part 2: Datatypes*, recommendation edition, May 2001.