Addis Ababa University
School of Graduate Studies
Faculty of Informatics

MDA Approach For The Development of Embeddable Applications On Communicating Objects

By

Eyob Alemu Buba

October 2005

A Thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

MDA Approach For The Development of Embeddable Applications On Communicating Devices

By

Eyob Alemu Buba

Name and Signature of Members of the Examining Board

No.	Name	Role	Signature
1	Dr. Dawit Bekele	Advisor	
2	Dr. Jean P. Babau	Advisor	
3			
4			

ACKNOWLEDGMENTS

I would like to express my gratitude to all those who helped me to complete this thesis. Special thanks are due to my advisor Dr. Dawit Bekele. He monitored my work and took effort in reading and providing me with valuable comments on major outcomes of this thesis work. In the face of the challenging nature of this work, his overly enthusiasm and integral view on research and his mission for providing only high-quality work, has made a deep impression on me. I owe him lots of gratitude for having me shown this way of research.

I equally wish to extend my appreciation and thanks to my external advisor Dr. Jean-Philippe Babau, with out whom this work wouldn't have been a reality. I also appreciate Julien De Antoni, for his dear concerns and honest comments through mails from France.

I also wish to register my profound gratitude to the MicroLink Information Technology College for the workload protection and partially financing my thesis work. The conducive environment that the college creates during my thesis work also deserves lots of appreciations.

I also extend my appreciation and thanks to my friends Zewdu Gebeyehu, Semahegn Abebe, Tamirat Tesfaye, Gutema Jirra and others who encouraged me during my work.

Table of Contents

List of Figures

List of Tables

ACRONYMS AND ABBREVIATIONS

ACL: Asynchronous Connectionless Link

API: Application Programming Interface

ASP: Application Specific Platform

CAN: Controller Area Network

CORBA: Common Object Request Broker Architecture

CWM: Common Warehouse Metamodel

DCOM: Distributed Component Object Model

GRM: General Resource Modeling

I2C: Inter IC Communication

IrDA: Infrared Data Association

IrLAP: Infrared Link Access Point

IrLMP: Infrared Link Management Protocol

J2EE : Java 2 Enterprise Edition

L2CAP: Logical Link Control and Adaptation Protocol

LAN: Local Area Network

LM_MUX: Link Management Multiplexer

LSAP: Link Service Access Point

MAC: Medium Access Control

MDA: Model Driven Architecture

MOF: Meta Object Facility

MTB: Maximum Transmission Boundary

OMG: Object Management Group

PAN: Personal Area Network

PDU: Packet Data Unit

PIM: Platform Independent Model

PM: Platform Model

PSM: Platform Specific Model

QOS: Quality of Service

RTOS: Real-Time Operating System

SAN: System Area Network

SCO: Synchronous Connection Oriented

SDU: Service Data Unit

UML: Unified Modeling Language

WAN: Wide Area Network

XMI: XML Metadata Interchange

XML: eXtensible Markup Language

ABSTRACT

Complexity is an ever increasing and inherent characteristic of software development. A major source of complexity in software development is technology, which produces a variety of implementation platforms that exist at the time of development and that arrive as future inventions. Changing existing software developed due to the growing variation of implementation platforms is becoming almost impossible. Hence a new development methodology called MDA (Model Driven Architecture) has been recently introduced with a strategy of separating the specification of the software system from the specification of its implementation on platforms as two different concerns of development. The two concerns are described as Platform Independent Model (PIM) and Platform Specific Model (PSM). MDA is now being successfully practiced as a promising solution at enterprise level software systems.

Recent technological advances are making possible the embedding of both processing and communication functions in highly integrated, low-cost objects such as PDA's and cell phones. This is promoting the use of a distributed approach in many application fields including embedded systems, which is now leading to the current and future realm of pervasive computing. The MDA success at the enterprise level has made it a viable choice for other domains that face a similar or even worse level of complexity such as the domain of embedded systems. However, recent efforts focused on extending the modeling capability of the core standards of MDA, particularly UML, towards the concepts in embedded systems such as Resource and Quality of Service (QoS). In addition, unlike the enterprise level platforms, there is no abstraction or middleware layer that can encapsulate all the variation in this domain that makes the variety of the platforms to appear as different choices. Therefore, adapting the MDA towards this domain requires a new approach that recognizes such peculiarities. Focusing on the communications subsystem, this work introduces an MDA based approach for the development of embeddable applications on communicating objects. A QoS aware and resource oriented approach, which exhibits the runtime interaction between applications and platforms, is proposed. The reservation based (typically connection oriented) networks are considered in this work. The applicability of the approach is also presented for Bluetooth and IrDA that shows the separation of application level reservation request from the actual network level reservation provided. We believe that this way the concerns of application level modeling and implementation could be separated from the platform level service specification and implementation as two different concerns of development in this domain.

CHAPTER 1. Introduction

The complexity in software development is beyond that encountered in other engineering fields. There is a constantly changing requirement, and a variety of implementation platforms. The field is very dynamic in that new technological inventions are inherent and very frequent that cause existing systems to suffer from small to large scale modifications. In some cases the change might go up to total system redesign. Heterogeneity of implementation platforms is currently acknowledged as unavoidable in the software engineering field.

Other fields of engineering are successful in managing product complexity through modeling for exploring the overall behavior of a product in its entire life cycle. Modeling enables to state clear definitions, well understood properties and the principles to the construction of products [1].

After the concept of modeling has been introduced to the software engineering field, it has opened a very important door for the handling of the growing complexity in software development. Although it is very difficult to model and predict all aspects and behaviors of a system in software development, an advantage of software engineering is that the modeled features can be standardized and can lead to automated construction of the products and the bridges/connectors between different implementations based on the defining models as well as automated regeneration for newly invented platforms [2].

1.1 Motivation

The OMG has recently introduced the Model Driven Architecture (MDA), which enables software to be developed beyond specific platforms. The MDA aims at increasing the reuse of existing designs, reducing the time of new developments, and supporting current and future developments. To achieve these goals, the MDA approach promotes a clear separation of the fundamental logic of the specification from the particular technologies that implement it through the key role of Modeling. It has been promoted in the enterprise information system development [2].

With the dynamics of the computer technology, new technological inventions arrive at a very fast pace. Most new technologies aim at making information availability to users as convenient as possible.

During the last two decades, technological advances in hardware made possible the embedding of both processing and communication functions in highly integrated, low-cost components, fostering the use of a distributed approach in many application fields including embedded systems. This fact led to the dissemination of so-called Distributed Embedded Systems (DES), which became the core of intelligent equipment with a high degree of autonomy, from robots to machine tools, from cars to trains and planes. Another class of DES appeared from the interconnection of consumer equipment, most notably computer peripherals and portable devices, such as laptops, mobile phones, PDAs and digital cameras, as well as within intelligent home systems, either for access control, location aware services, security and distributed multimedia [3].

Pervasive computing, ad-hoc networks, mobile computing, and sensor networks exhibit the current importance of networking within embedded systems. The networks makeup the most important platform service for DES and pervasive systems.

Generally, the embedded platform elements are classified in to three basic subsystem components, i.e., the OS, the Communication subsystem, and the I/O Device Driver subsystem three different but integrated system resources [16, 10].

Today, a large variety of networks are currently available to build distributed embedded systems. This makes the platform variability significant in the domain of small devices and embedded systems. All the three subsystems exist in many different variations. There are many operating system and network choices. Most of them are competing in the same domain of application. For example, CAN and TTP in automotive systems, IEEE 1394 (FireWire) and USB for multimedia devices and peripherals interconnection, Ethernet and Wireless LAN for office automation and now for industrial automation and also for multimedia devices and peripherals interconnection, Bluetooth, based on radio, and IrDA, based on infrared light, to interconnect peripherals and portable devices. Future requirements of pervasive systems will demand the interaction of applications deployed on all objects that may use any of the above and other networks [48].

In this work, we used the term ***Embeddable Applications On Communicating Objects*** to describe an environment, which enables applications to communicate using the available network. This could be possible if their design and also implementation on platforms is not dependent on a specific network. As long as there is an appropriate intermediary between the applications and the target network, applications that may have some information to share with each other, can easily communicate even though they are developed for different purposes. This is one future vision of pervasive systems development.

MDA is not a modeling language or standard by itself. But it is a methodology (reference model) that integrates a set of modeling tools and standards for a new development approach. Although it has been introduced and promoted in the enterprise systems, its concepts, goals and benefits make it very interesting to be adapted to other domains of system development that face similar or even worse levels of complexity such as the domain of embedded systems and more importantly the domain of pervasive systems since it makes extensive use of embeddable communicable devices and applications.

1.2 Statement of The Problem

As stated above, MDA has been used at the enterprise level as the future's successful software development methodology. At this level, many of the platform artifacts such as the used programming language, operating system, network and other resources are abstracted by the middleware technologies so that the approach considers only the middleware architecture variability.

However, this is different in embedded systems, where most platform elements are openly available either at the API level or directly at the hardware level. The network, being one subsystem in the embedded system platform, is presented to its users through a corresponding API. Since there are several networks that can be possibly used by embedded systems such as Bluetooth, IrDA, HomeRF, HyperLAN, Ethernet, WLAN, CAN, I2C etc., a number of APIs exist.

The other essential aspect is Quality of Service. The limited resources at this level make the systems to be QoS sensitive for using the available resources effectively. Almost all networks used at this level are QoS aware. But, the services provided by each interface, the QoS provisions and parameters used to define them are different. This creates network level variability.

As future applications are becoming pervasive and embedded in small communicable devices, this network level variability must be handled in a formal development methodology. To make embedded application design easier and more portable so that they can target different available networks, the development of the applications should start from a point, which is independent of a specific network.

As described in [16,17], modern embedded systems have certain characteristics that demand new approaches to their specification, design and implementation. They are being developed as a composition of subsystems and functional units that carryout responsibilities of computation and

communication using possibly a heterogeneous set of subsystems. The traditional monolithic design and implementation is being transformed to a new paradigm. The emphasis is given on two aspects: early modeling of the systems and methods to delay the commitments to particular components or implementations [17].

So far, there is no work that has addressed this network level variability with an MDA approach. Most of the works so far focus in extending the MDA key standards such as UML towards addressing the modeling concepts in the embedded domain. This work proposes a platform based MDA approach for the network subsystem of embedded platforms, which includes a generic network interface model and a mapping methodology towards a specific network model. By platform based, we mean that the approach will focus on platform abstraction towards applications instead of application design and refinement towards platforms.

1.3 Research Objective

The general nature of embedded applications is their QoS requirement, which results from the need to effectively utilize the limited capabilities of the target platforms. In this aspect, most target devices and specifically the network platforms for embedded applications share the same limitation. This work will propose an MDA approach that can support model based QoS enabled development of future embedded applications on communicating devices that can target many different networks.

The main objective of this research is to propose a possible adaptation of MDA to the development of communicable embedded applications focusing on services of the communications subsystem of the platforms. The approach is based on the platform based development initiative proposed in [16,17].

The Quality of Service issue is primarily considered and an appropriate modeling and transformation approach is proposed. There are a number of QoS characteristics and categories related to embedded systems and networks in general. The major categories are Performance, Dependability, Coherence, Throughput, Latency, Efficiency, Demand, Reliability, Availability and Security as stated listed in [9,10].

Performance is a key issue in embedded systems. In this work we will be dealing with the performance category, which incorporates Throughput and Latency.

The specific objectives of the research work are:

- An approach for modeling the services and the QoS provisions of specific network platforms. In this approach, what elements the model should include will be proposed. This is called the Platform Model (PM) in MDA.
- An approach for modeling the platform independent services and QoS specifications of a generic network interface for embedded communication. This will be a means for applications to specify their required services and QoS requests to an abstract platform.
- An approach for mapping the platform independent model with a specific platform model. This includes both services and specially the QoS mapping. Since the PIM is interacted by applications, it is considered as a specification of the required QoS. On the other hand, the Platform Model (PM) describes the characteristics of a specific network, and it is considered as a specification of the provided QoS. The mapping layer has a responsibility of verifying the provided and the required QoS and transforming the description in the PIM to an appropriate description to the PM target thereby creating the Platform Specific Model (PSM).

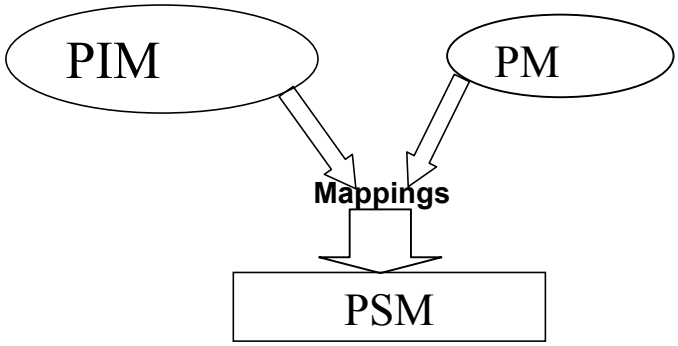

Figure 1-1: The general MDA process

1.4 Scope and Limitations of The Research

As detailed in chapter 2, it has been realized that the embedded networks can be classified in to two broad categories based on their QoS mechanisms: Reservation based and Priority based. The work will focus on the Reservation based category where typically connection oriented service is provided and controlled medium access mechanism is employed. The Priority based category, where typically a connectionless service is provided and an uncontrolled medium access mechanism is employed, is not covered by this work.

The approach tries to consider both the Design Time (analysis) and Run Time (verification and mapping) aspects of the PIM versus PM mapping mechanism between the platform independent and the target platform model.

Implementation aspects and code generation are not covered in this work.

1.5 Methods and Procedure

The first activity has been background study and review of related works in which a number of works have been studied and analyzed. Some of them have provided the basis for this work.

Below is the procedure, which shows the intended methodology of the major works and the flow of activities. The course of action starts from the study and analysis of concrete networks and continues modeling the abstract level (PIM) and finally completes by devising a mapping strategy that links the two levels.

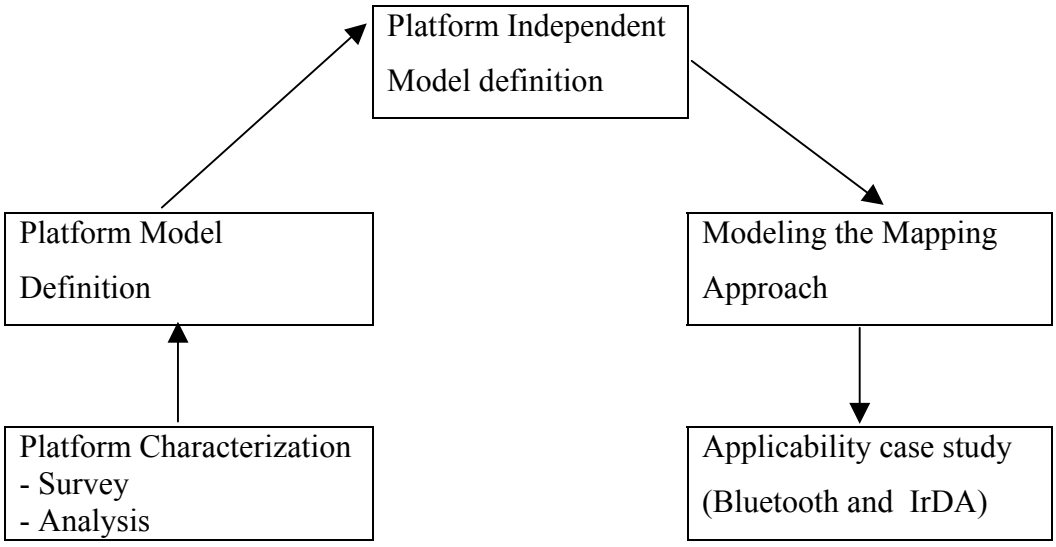

Figure 1-2: The basic activities and procedure

Analysis of the embedded networks

The major service and QoS provision characteristics of the networks have been defined with this activity. Performance QoS category is the major issue addressed in this work (throughput and latency). The major factors and parameters that describe these performance elements are identified.

Model development

After the modeling elements are determined, the static and dynamic aspects of the networks are described according to the UML concepts. The modeling elements for Platform Models are defined that can characterize all network platforms in the selected domain. Next, the corresponding PIM that can appropriately abstract the services and QoS aspects of the platforms is defined. We have tried to set up reasonable gap between the two models (i.e., the PIM and the

PM). After this, the appropriate mapping strategy that can map both the structural and the QoS models of the PIM and the PM to produce the PSM is modeled.

Applicability Test Cases

Finally, the complete approach will be evaluated and verified with a case study of two specific networks namely Bluetooth and IrDA.

1.6 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 will introduce the basic concepts that are fundamental for understanding the rest of the document and the work done. First, it will discuss about MDA and the concepts and its building blocks as well as it impact and success in software development. It will also discuss about basic characteristics and challenges in embedded systems and application development in this domain in relation to the future visions of pervasive computing. It then gives an overview of some example networks and a comparison between them. Finally, it discusses the QoS and the UML profiles that are used as a basis for defining the modeling artifacts in this work.

In Chapter 3, related works are discussed and reviewed. It lists some works done relevant to this work which have contributed concepts and frameworks that are used as basis in this work.

In Chapter 4, the main work of the thesis is presented. It starts by an overview and justification on the proposed approach. It then continues with the detailed analysis and modeling of the platforms focusing on QoS. The proposed PIM and mapping approach are also incorporated. Finally, the proposed model that indicates the conceptual and logical relationship between the three levels (PIM, PM, and Mapping) is presented.

In Chapter 5, the applicability of the approach by an application to the two typical reservation based networks, namely Bluetooth and IrDA is presented.

In Chapter 6, a general summary, conclusion and possible future works are presented.

CHAPTER 2. Background

2.1 Model Driven Architecture (MDA)

In 1997, the OMG has adopted the UML as a standard modeling language to be used as a modeling tool for object oriented software systems. Since then UML has gained popularity and support from the industry and the tool vendors. UML has contributed a lot to software engineering in a number of ways. It is programming language independent so that a design model in UML can be automatically transformed to code for a number of programming languages using a modeling tool. More importantly, UML has reduced the problem in the interaction between the stakeholders of a system. Use cases and interaction diagrams have helped users to understand the analysis models prepared by system modelers.

As the scope and diversity of software systems grows, a number of systems appear to be deployed on different platforms (such as OS's, Communication protocols, and programming languages) so that integration between them becomes a problem. This problem has been solved by the emergence of middleware platforms. A middleware has abstracted away a number of heterogeneous specificities through standard interfaces and high-level communication protocols. Because of this, system design models in UML focus on more abstract system specifications at the middleware level. In the traditional Object Oriented Software methodology the design phase reflects platform artifacts [58].

The recent challenge has been the existence of different middleware platforms. Systems with worldwide scope could be developed (designed and implemented) using any of the available or the appropriate middleware platform at hand. But, this created platform heterogeneity at the middleware level. From the recent developments, it has been observed that the middleware level heterogeneity and evolution will be unavoidable. Hence, what is envisaged by the OMG is a new development paradigm that can accommodate the heterogeneity at middleware level. This gives birth to the MDA methodology where platform independence is expressed by modeling systems free of the specific artifacts and concepts used in any of the middleware platforms.

The Model-Driven Architecture (MDA) is a design approach that creates a separation of concerns between the specification of a system and the implementation of this specification on a particular platform [18].

2.1.1 MDA Vision and Standards

The vision of MDA is a development environment in which efficient and nearly seamless interoperability between diverse applications, tools and databases is achieved through the interchange of shared models. Components participating in this environment leverage standard services provided by implementations of MDA standards that enable them to expose and interchange their metadata as instances of well-defined models. The platform services have standard definitions that are expressed via standard programming models (APIs), which are automatically generated from Platform Independent Models [3].

With the currently working MDA specification, it is possible to have an architecture that will be language, vendor and middleware neutral. Platform independent applications built using MDA, may range from transportation to health care industry and can be deployed on a range of open and proprietary platforms, such as CORBA, J2EE, .NET. [2,3,4].

As an example, let us consider a formal definition of an operation that transfers funds from a checking account to a savings account. The fundamental functionality of this operation is that a specified amount is subtracted from a designated checking account and added to a designated savings account, with a constraint that the two accounts must belong to the same customer. This functionality remains invariant regardless of whether the operation is performed by a CORBA, an Enterprise Java Beans, or a DCOM object.

Thus, a Platform Independent Model (PIM) is a formal specification of the structure and function of a system that abstracts away technical details. Hence, a DCOM specification of the funds transfer operation would be platform-specific, where the platform is DCOM. A specification that depends on interfaces and artifacts of CORBA would be another example of a Platform Specific Model.

There is also another model used in MDA, the model that defines a specific platform, which is used to transform the PIM to a PSM. This model is called a platform model (PM). An example is a Platform Model that describes the CORBA platform. When a platform specific model of a software system is generated, it would modify the PIM according to how it has to be realized on the target PM.

The key standards that make up the MDA include:

Unified Modeling Language (UML): which provides a graphical language for model specifications. It is used for designing models in both PIM and PSM. The UML standard is exclusively used in software systems modeling.

Meta Object Facility (MOF): which provides an extensible model driven integration framework for defining metadata and data in a platform independent manner. It provides the core definitions of the MDA and is placed as a root for the other related standards and modeling languages including UML.

XML Metadata Interchange (XMI): which provides rules by which transmissible schemas are generated to be exchanged between different tools and teams.

Common Warehouse Metamodel (CWM): this is an example for a domain based MOF for data warehousing systems which is set of standard interfaces that can be used to enable easy interchange of warehouse and business intelligence metadata between warehouse tools and repositories.

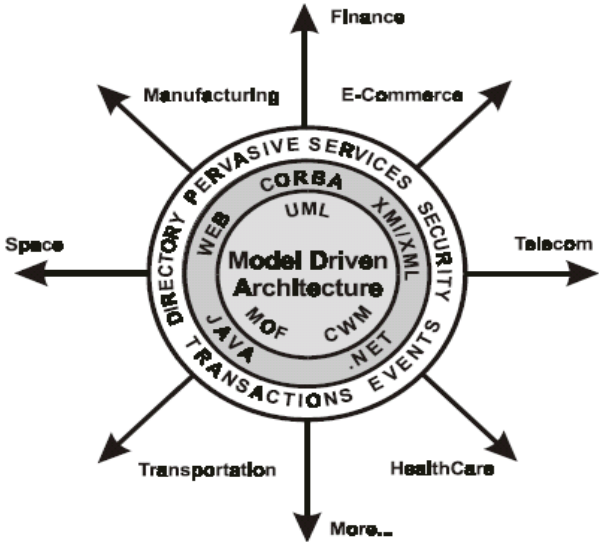

Figure 2-1: The Model Driven Architecture standards, visions and application areas

2.1.2 The MDA Process

According to [25], the process of MDA revolves around three major steps:

1. First, a model a system with higher level of abstraction is built. This model is independent of any implementation technology as Platform Independent Model (PIM).
2. Next, this PIM is transformed into one or more Platform Specific Models (PSMs). These PSMs are the constructs of PIMs into more detailed, platform dependent and technology oriented models for a particular platform such as .Net (DCOM) or Enterprise Java Beans (EJB).

3. Finally, the PSM is transformed into the code according to the programming languages supported by the platform. The PSM already carries all the necessary details, which are required for the coding.

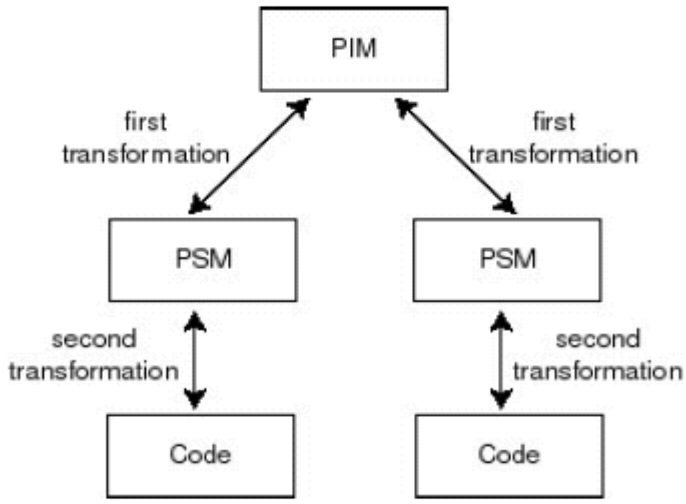

Figure 2-2: The MDA usage pattern

The above procedure is currently used in the development of general software systems.

2.1.3 Abstractions and Viewpoints

The major concept behind the objective of MDA is Abstraction [18]. In the development process of a software system, there is a need to consider many aspects such as the hardware the system will run on, the other systems it should be able to interact with, and so on. Incorporating all these aspects into a single model results in a complex model, even though it may be useful. A better choice is to identify models in terms of the abstraction criteria that are used to extract what is included in the model at a higher level. A model created based on specific abstraction criteria is often referred to as a model from that **viewpoint**, or in short as a ***view*** of the system.

Another common use of abstraction is to omit details from a model. In contrast, creating the lower level details from a higher-level abstraction is often called refinement. Each element of a higher-level model may be mapped to one or more elements of a lower level refinement.

PIMs and PSMs can be taken as views of a system at different abstraction levels.

2.1.4 Models and Metamodels

A model is virtually any collection of expressions produced using a language that has well-formed notation (syntax) and meaning (semantics). A more precise definition of a model, as used in the Model Driven Architecture [18], is:

"A representation of a part or the complete functionality, structure and/or behavior of a system, as seen from a particular viewpoint, described in a language which is well-formed in syntax and semantics"

Examples of models are architectural blueprints of a house, C++ source code, UML diagrams, and XML document type definitions.

Besides the concept of models, the Model-Driven Architecture uses the concept of **Metamodels**. The formalism of a modeling language is defined in a metamodel. Metamodels specify the syntax and semantics of models. In the same way, metametamodels can be used to represent the syntax and semantics of metamodels. In this specific application of metamodelling, a model can be seen as an instance of a metamodel, while a metamodel can be seen as an instance of a metametamodel. This 'instance-of' relationship can be applied as often as necessary, creating as much metalevels as needed. However, MDA defines only four layers that are organized in such a meta relationship among them [24][19], creating the hierarchy shown in figure 2-3. The OMG has adopted for the Model-Driven Architecture as a metamodelling layered structure in which the highest level (the metametamodel) is described and defined using the Meta-Object Facility (MOF). The Meta-Object Facility is reflective in that it is defined using itself, applying a small set of core constructs. The models at metamodel (M2) level define an abstract language that is used to create models at (M1) level. The number of constructs increases from level M3 to level M1. As shown in figure 2-3, there are more UML (M2) elements than MOF (M3) elements.

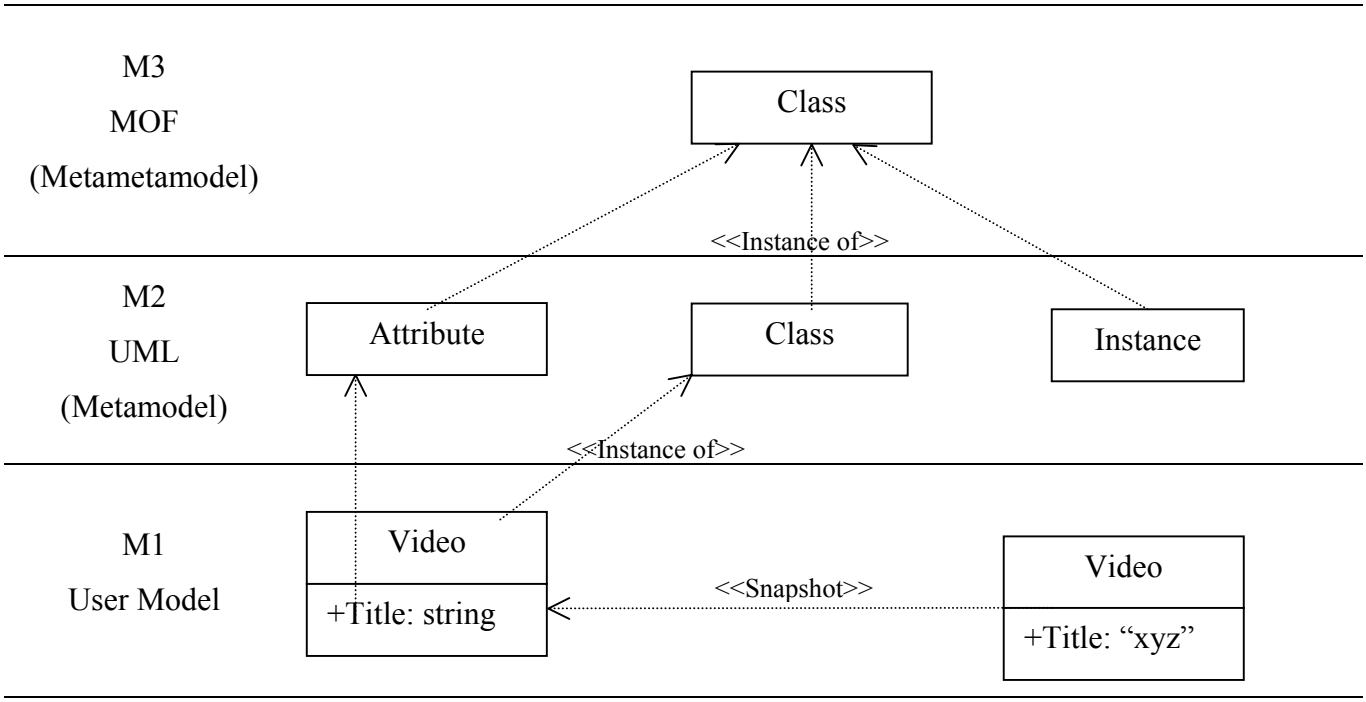

Figure 2-3: An example of the four layer Metamodel hierarchy in MDA

For defining models at metamodel level, there are two options. One choice is to define a completely new metamodel based on the MOF constructs as it is done for CWM in contrast to UML, or extending an existing metamodel to provide the desired modelling capabilities. The UML profiles are examples of the of UML extensions defined for different application domains such as real-time and embedded systems and Quality of service.

2.1.5 Transformations

One of the fundamental ideas of the Model-Driven Architecture is to provide the ability to support software throughout its entire lifetime. For example, whenever new platforms emerge, one has to decide whether or not to adopt this new platform and deploy existing software on it. In addition, not adopting this new platform may lead to considerable loss of investment or income in case the old platform becomes obsolete. Ultimately, developers want to obtain a condition in which the costs of porting current software to a new platform is considerably lower if the new platform is to be adopted as a potential target. This scenario is one example of the key problems that the Model-Driven Architecture is addressing, platform variability and evolution. The use of model transformations would reduce the investments and time required to port the software to new platforms.

Model transformation refers to the process of transforming (relating and mapping) elements of one model into corresponding elements of another model. Model transformations are usually

defined at metamodel level (M2) and applied at model level (M1). Sometimes they are also applied at the metamodel level (M2). The definition of model transformations requires knowledge of the metametamodel level (M3) and, potentially, other information required during the transformation.

[3, 18] propose the usage of OMG core technologies to express model transformations, model markings and annotations. However there is no standard mechanism for defining model transformations and model markings. The OMG has issued an RFP (Request for proposal) for the design of the Query/View/Transformation specification [20]. This specification is expected to provide a standard mechanism for obtaining information on models and defining transformations using the Meta-Object Facility. Figure 2-4 shows the metamodel for MDA [26].

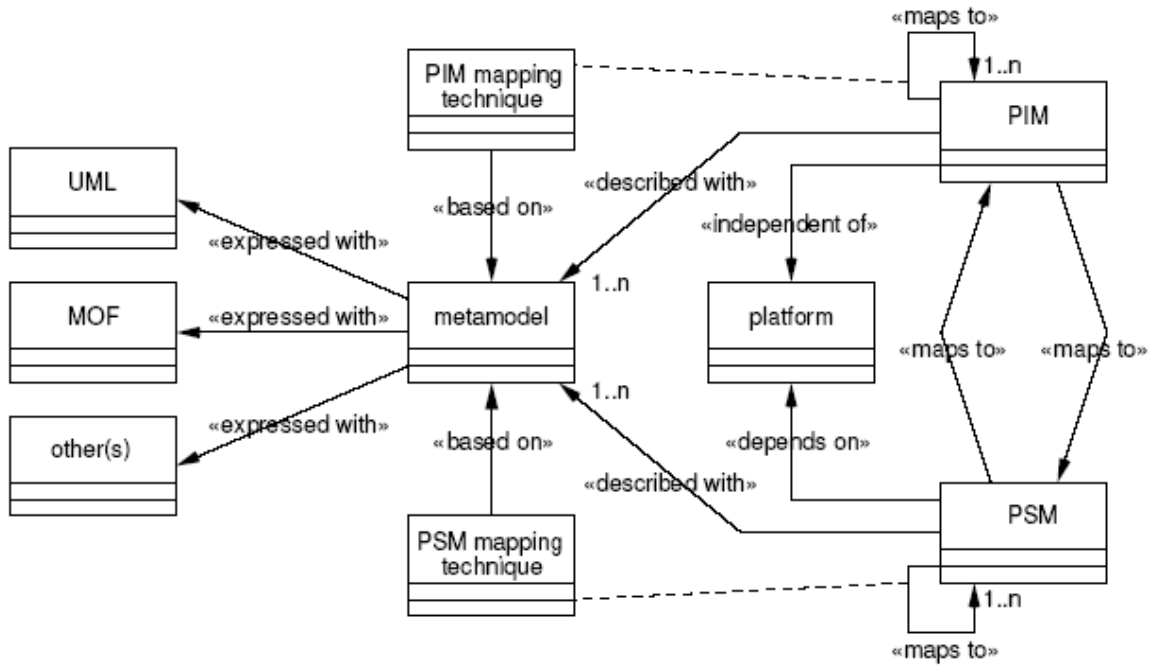

Figure 2-4: The Model-Driven Architecture metamodel.

2.1.6 MDA Success

With this strategic approach, MDA is gaining wide range acceptance and usage in the software development community in which the OMG is playing the central role in standardizing the MDA modeling definitions and specifications. Many generic and domain based modeling specifications have been released. Some software development companies and business organizations are using the MDA technology as a new and successful approach and in their system development process

and they are witnessing its advantages. This can be seen as a sign that the MDA will be the future's dominant development technology in the field of software engineering [27].

2.2 Development of Embeddable Communicable Applications

In the past, the term embedded had been used to describe specific, hardware related systems plugged in many computing and non-computing devices that range from small hand held to large complex electronic instruments [6]. This hardware dependence requires developers to spend much time in studying and resolving hardware/technology problems instead of focusing on application requirements and design. Even in some cases the development process is done through the use the co-design scheme for both the hardware and software at the same time. This has been a major source of complexity and luck of portability in this domain.

Another typically faced problem in the design of embedded systems is the lack of unified engineering approach [14]. So far, there is no agreed or standardized development model for embedded systems. The close link between the embedded software and the host hardware has influenced the combination to be termed as an embedded system [6].

However, in recent years, the capability of the hardware devices is enhanced to provide extensive interfaces and the possibility of hosting applications of deferent types. And this has introduced new development mechanisms for embedded applications in that the stringent hardware dependent constraints can be relaxed and specified with some acceptable range and hence this opens the possibility of designing the applications independent of a specific target platform. Programmable interfaces and software abstraction layers are becoming possible to support flexible system developments.

This evolutionary enhancement of embedded systems from their specific purpose functionality to a more general, multipurpose and more intelligent capability is making the devices not only capable of hosting embedded applications but also communicate with each other to share resources and to transfer information [7]. In pervasive systems development, embedded systems and applications play a vital role [34]. If a formal methodology can be defined for the application development in this area, the future vision of pervasive computing in deploying communicable applications on all communicating objects could be attained.

Pervasive systems development can be seen from two perspectives from the global (wide area) perspective and local area perspective. On the wide area perspective, large-scale networks including the Internet will be used as the communication medium between the interacting applications. In contrast, on the local area and small-scale perspective, the communication

networks are usually the short range, low level networks stated in chapter 1. Often, the local area systems are interconnected with the external wide area systems through standard gateways such as OSGI (Open Systems Gateway Initiative) and UPnP (Universal Plug and Play).

Finally, there is a general fact that the capability of resources in the embedded domain is much smaller than that of desktop systems. The platform architecture determines functionality and the capacity determines the quality of service provided for applications. Therefore the functionality and the quality of service achieved by applications are highly dependent on the embedded system platforms.

2.2.1 The Embedded system platforms

The term platform is in general defined by OMG [2, 26] as:

> "*Technological and engineering details that are irrelevant to the fundamental functionality of a software component.*"

In [15], a more precise definition for an embedded platform is presented as a "family of Micro-architectures possibly oriented towards a particular class of problems". As embedded development is very complex, the platform based approach proposed in [16] and further improved in [15] has significant impact in order to formalize the development process and simplify the complexity. Using Platform Based Design approach, the platforms for embedded systems are modeled at different abstraction levels so that developers could choose the appropriate abstraction level that can avoid their concern about the details of the platforms. A typical architecture of an embedded platform is shown below (Figure 2-5) [15].

<table>
<tr><td colspan="3">Application Domain specific Services
(Functions, User Interfaces)</td><td>ASP
platform</td></tr>
<tr><td>RTOS</td><td>Network
Communication
Subsystem</td><td>Device Driver</td><td rowspan="2">API
platform</td></tr>
<tr><td>Microprocessor
and Memory</td><td>Interconnection</td><td>HW, I/O</td></tr>
<tr><td colspan="3"></td><td>ARC
platform</td></tr>
</table>

Figure 2-5: Platforms at different levels

As shown in the figure, there are three abstraction levels: architecture (ARC), application programming interface (API), and application specific programmable (ASP) Platforms. The ARC Layer includes a specific family of micro-architectures (physical hardware). The API Layer is a software abstraction layer wrapping ARC implementation details. API presents what kinds of logical services are provided and how they are grouped together and represented as interfaces. For example, it may show that data transmission is supported by a communications subsystem, but not how this service is implemented with in the hardware. For such purpose, RTOS, device-driver and network communication subsystem are treated as three components of the entire platform [15].

ASP is a platform, which provides a group of application domain-specific services directly available to users. In addition to calling these existing services, users sometimes also need to modify or combine them, or even develop new services to meet certain requirements.

2.2.2 The communication subsystem of embedded system platforms

According to the ISO/OSI strategy, a network is organized in to 7 different layers each assigned with different functions and services.

Generally, there are two major types of networks depending on size, namely LAN and WAN although there are some other types which are special cases of these categories. For example PAN (Personal Area network) and SAN (System Area Network) are special cases of LAN. The required layers for transferring information through the network depend on the size of the network. Accordingly, LAN networks require the bottom two layers of the network. On the other hand, WAN network require the upper layers for routing between different networks.

Every layer has a definition in terms of the services it provides to its clients (upper layer protocol entities or applications). A layer provides its services through service primitives. Upper layers must have access to these service primitives in order to deliver and receive their data (to send and receive information from peer entities with which they communicate). Such a set of primitives that provide the services of a layer is called an interface of the layer or a Service Access Point (SAP). For higher layers (i.e., users of a given layer) this is an abstraction of all the underlying (lower) layers of the network.

When we look at the fundamental definitions of the layers, the transport and the data link MAC layer specifications of the OSI have relatively similar objectives. The major objective of the MAC layer is providing a reliable data transfer between peer nodes because the data traverses the physical medium between the two nodes that could be unreliable. On the other hand, the transport layer provides a reliable communication between two application/process end points whose communication may traverse several networks using the relatively unreliable network layer routing services [1].

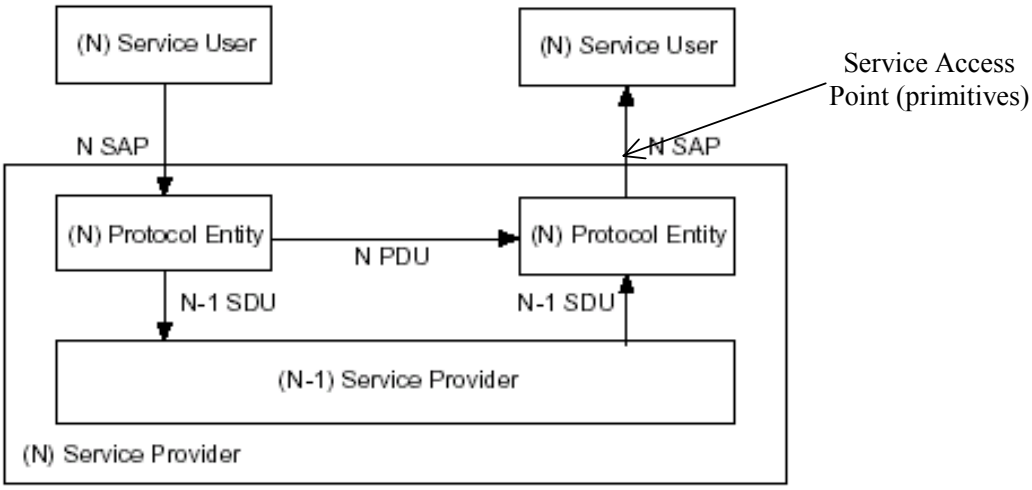

Figure 2-6: A layer (layer N) and its Service User (Upper layer / Application)

When we consider embedded networks, they are typically LAN networks. Some are very small range described as Personal Area Networks. Their specifications basically emphasizes on the Link Layer Services. The upper layer services are specified as optional and they are implementation dependent. Some embedded networks such as CAN have only the Link layer specification and leaves the upper layers open for any kind of implementation. Other networks such as IrDA have two categories in their specification. It specifies the **Required** category, which includes all link layer protocols and services, and the **Optional** category, which includes all the rest of the protocol layers.

From the above facts, it can be observed that in LAN, and typically in embedded networks, the use of the upper layer protocol can be avoided and applications can directly communicate with the link layer services so that an unnecessary protocol overhead is avoided. This is important because routing services of the network layer and the additional services of the transport layer are not needed. This eventually will improve the performance requirements of the network, which is essential in embedded system development.

The network/application interfacing mechanism could be possible with a design methodology that enables an application (or higher level system) to request, negotiate, and make a contract with the underling network. To realize this, a scenario where direct interaction between applications and the underlying network infrastructure is definitely needed particularly in the case of embedded communication. Hence, the need arises to define programming interfaces suitable for QoS-aware applications deployment on target short-range networks. According to these models, applications can request communication services with preferred bounds on communication throughput or end-to-end latency [58].

Considering that there are several target networks with unique interfaces for the applications, an MDA approach will provide an enabling methodology in making the application development process network neutral. This enhances the chances of retargeting capability of the applications towards different networks. Using models to define functional interfaces and related QoS, both a generic (PIM) and concrete (PM) models can be strategically produced to address the variety of communication platform elements.

For our purpose, we take the mandatory (required) layers of the embedded networks to define the platforms. As described above the mandatory layers of the networks are the link layer protocol layers.

Therefore, we will define the link layer functional and QoS interfaces of the networks as platform models (PMs). Based on this, a generic interface that can be taken as a PIM is modeled and then a mapping model that can transform the generic interface to a specific interface is defined.

2.2.3 Example Networks

A. Bluetooth

Bluetooth is a short-range radio link initially intended to replace the cable(s) connecting portable and/or fixed electronic devices. It utilizes a slotted channel applied with a nominal slot length of 625 μs, which divides every second into 1600 slots. For full duplex transmission, a Time-Division Duplex (TDD) scheme is used. On the channel, information is exchanged through packets. Each packet is transmitted on a different hop frequency. A packet nominally covers a single slot, but can be extended to cover up to five consecutive slots [28].

In Bluetooth, the used link type makes the basic QoS differentiation. There are two link types SCO (Synchronous Connection Oriented) and ACL(Asynchronous Connectionless Link). The SCO link provides a synchronous data link connection for time-bounded voice only data, while ACL provides connectionless transfer. The ACL can be parameterized with different QoS parameters for varying levels of QoS.

For ACL, the Bluetooth user QoS is defined at the L2CAP (Logical Link and Adaptation Layer). This layer also monitors the utilized resources (on interacting devices) and ensures that QoS contracts are honored. The L2CAP allows QoS negotiation between Bluetooth terminals by using a QoS flow specification.

Since the lower baseband layer creates a single ACL link, the L2CAP performs multiplexing of several flows where each flow can have a separate QoS configuration.

A master of a Bluetooth piconet (the smallest group of a bluetooth interconnection composed of up to 8 stations) always starts transmissions at even numbered timeslots. A slave (one of the hosts other than the master) is able to respond to the master using an odd numbered slot if it has been addressed in the preceding even numbered slot.

The master of a piconet, schedules the transmission opportunities. For SCO links, the timeslots are reserved. In this case, a slave is able to transmit on the reserved slot regardless of whether it has successfully received a master transmission during the previous slot. But only up to 3 SCO links that support 64 Kbps are supported appropriate for voice traffic.

For controlling the transfer of different L2CAP flows, only a single ACL link is available at the lower baseband layer between a master and a slave terminal. The QoS of the ACL link for data transfer is defined by parameter called **poll interval**. The poll interval defines the

maximum duration between subsequent transmissions from the master to a slave on the ACL link.

In summary, there are two major link layer protocols in Bluetooth, namely the Baseband and the L2CAP. While Baseband provides reliable data transfer using retransmission and error detection, L2CAP is responsible for providing the link layer services to upper layers.

B. IrDA

IrDA is a short-range infrared data communication standard. IrDA stands of Infrared Data Association, which is an industry-based group of over 160 companies that have developed communication standards especially suited for low-cost, short-range, point-to-point communications at a wide range of speeds [29,30].

The IrDA usage model is for short-range directed communication link that supports ad-hoc point-and shoot type of communications. The nominal operating range is a 1m cone with 15 degree half-angles.

IrDA provides half duplex transmission mechanism by which the two communicating devices will take turns to send data to each other. Each device has to wait 10 – 500 ms for the other part to finish its turn.

One characteristics of IrDA is that the effective data rate is the negotiated rate between the communicating devices at link setup time. Furthermore, there can only be two devices communicating in a single session where one device is called primary and the other device is called secondary.

In a typical situation, one device is sending data to another. The sending device must allow the receiving device to briefly own the link between each burst of data. This process is called link turn around.

The important link layer protocols in IrDA are the Link Access Protocol (IrLAP) and the Link Management Protocol (IrLMP). Both are required IrDA protocols corresponding to OSI layer 2 (data link layer). While IrLAP provides reliable data transfer using Retransmission and Error detection**,** IrLMP is responsible for providing the link layer services to upper layers.

C. CAN

CAN is a serial bus system especially suited for networking devices such as sensors and actuators within a system. CAN is a serial bus system with multi-master capabilities, that is, all CAN nodes are able to transmit data and several CAN nodes can request the bus

simultaneously. CAN protocols cover the lowest two layers of the OSI reference model. It is widely used in automotive and industrial technologies. Higher-level protocols are utilized by applications and operating systems that can properly interface the MAC layer CAN network [31].

In CAN networks, there is no addressing of nodes or stations. Instead, prioritized messages are transmitted. Every message has unique identifier throughout the network that is maintained at the network design time. The identifiers also tell the priorities of the messages from each node that cannot be changed dynamically. The identifier with the lowest binary number always has the highest priority.

A transmitter sends a message to all CAN nodes by broadcasting. Each node decides on the basis of the identifier received whether it should process the message or not. The identifier also determines the priority that the message enjoys in competition for bus access. Competing nodes arbitrate through the message identifier using bit-wise arbitration. Unlike CDMA/CD this arbitration is loss less.

D. I2C

I2C is a network initially developed for interconnecting Micro controllers and Integrated Circuits. I2C stands for **Inter IC Communication**. In this network there may be one or more Master nodes that control the network usage. A master is a node (device) which has the capability of initiating a data transfer on the bus and generates the clock signals to control and synchronize the transfer. At that time, any device addressed is considered as a slave. All the rest of the nodes are also called slaves that can only wait for signals from a master and respond. The master node is responsible for starting a data transmission, controlling the transmission, and finally terminating it. Typical I2C served devices are devices with Micro Controllers, LCD drivers and Memory Interfaces. Depending on its capability a device can be either a master, a slave or can act as both [32].

A master selects or polls a target slave with which it wants to communicate by sending a poll frame. The polled slave responds and the communication begins between the master and this slave node. Masters themselves can be slaves if they are polled by another one. Only two nodes can communicate at a time even though there may be many devices connected to the I2C bus.

The I2C bus can also support multiple masters. If there are more master nodes in the network, they may contend for the channel when they start transmission simultaneously. The winning

master takes control of the channel through bit-wise arbitration similar to that of CAN. Although there are no predefined massage priorities in I2C, the arbitration takes place on all message frame bits until one random master eventually wins. There is no limit on the data size in I2C but following every byte of information, an acknowledgment bit is returned to the sender. Every message starts by a **Start Condition** and terminates by a **Stop Condition.**

A summary of the relative architectural alignment of the protocol layers of the above networks is summarized in Table 1 and a comparison is between them is presented in table 2-2. I2C and CAN define only the lower two layers of the OSI model.

<table>
<tr><th>OSI Layer</th><th>BlueTooth</th><th>IrDA</th><th>CAN</th><th>I2C</th></tr>
<tr><td>7 – Application (user services)</td><td>File Transfer Profile (FTP)
Generic Object Exchange Profile (GOEP)</td><td>Point and Shoot Profile (PnS)</td><td></td><td></td></tr>
<tr><td>6 – Presentation (data format, encryption)</td><td colspan="2">Object Exchange Protocol (OBEX)</td><td></td><td></td></tr>
<tr><td>5 – Session (session management)</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4 – Transport (error recovery, flow control)</td><td rowspan="2">RFCOMM
Service Discovery Protocol (SDP)</td><td rowspan="2">Tiny TP
Information Access Service (IAS)</td><td></td><td></td></tr>
<tr><td>3 – Network (switching, routing, addressing)</td><td></td><td></td></tr>
<tr><td>2 – Data Link (encoding/decoding, media access control)</td><td>Logical Link Control and Adaptation Protocol (L2CAP)
Link Management Protocol
Baseband</td><td>Link Management Protocol (IrLMP)
Link Access Protocol (IrLAP)</td><td>Message filtering
Acknowledge-ment</td><td>Arbitration
Acknowledge-ment</td></tr>
</table>

OSI Layer	BlueTooth	IrDA	CAN	I2C
1 – Physical *(signal)*	Radio Hardware	Infrared Controller, Transceiver	Signal level Bit representation	Signal level Bit representation

Table 2-1 The OSI layer and the relative alignment of the embedded networks

The shaded area shows, the relevant layer for embedded systems development.

Property	**IrDA**	**Bluetooth**	**CAN**	**I^2C**
Bandwidth	**Ver.1**: 115.2 kbps **Ver.2:** 1.152 Mbps **Ver.3:** 4 Mbps **Ver.4:** 16 Mbps	1 Mbps	1 Mbps	**Standard:** 100Kbps **Fast:** 400 Kbps **High speed:** 3.4 Mbps
Access Mechanism	**NDM**: Listen for 500ms (**Contention**) **NRM**: Master/Slave Primary station passed token for 500 ms (**Controlled**)	**TDMA:** Polling scheme – negotiable for QoS; Master/ Slave based. Master polls Slaves (**Controlled**)	CDMA/CD with message priority. (**Uncontrolled**)	Similar to CAN bit-wise arbitration on Frame bits (No predefined message priority. (**Uncontrolled**)
Maximum Data size per Message	160-2048 Bytes (within 500ms)	2-342 Bytes	8 Bytes	Not Limited, But each byte must be acknowledged
Built-in QoS Provision and management	Negotiation, **Reservation**	Negotiation, **Reservation**	**Priority**	**Priority**
Transmission Method	Unicast, point to point **Connection**	Unicast, Multicast through master **Connection**	Broadcast, **Connectionless**	Broadcast, **Connectionless**

Table 2-2: A comparison between the networks

2.3 Quality of Service

In this section we deal with general QoS concepts. The term Quality of Service has originated from the domains of Distributed Systems, Networks and Embedded systems. Quality of Service is defined in [9] as:

> " *A set of perceivable characteristics in user friendly language with quantifiable parameters that may be subjective or objective.*"

Subjective qualities are usually observable and objective qualities are usually measurable. Quality of Service generally covers system performance, as opposed to system functionality. QoS requirements specify not what the system does, but how the system satisfies its clients while doing what it does [42].

The characteristics of Quality and their parameters are expressed in user-friendly language on two types of concerns.

User (client) Satisfaction: this is based on the stated user requirements

Resource Consumption: for the provider of the required service to mange and control the consumed resources.

The QoS relationship between the requester and the provider can be viewed from two aspects:

- **From Client/Server (horizontal relationship**) in which case a client specifies the required QoS and the server specifies the offered QoS and then they agree with a negotiated QoS contract.
- **From an Abstraction/Realization relationship (vertical relationship)** in which case the relationship is seen in a layered architecture. Top level (abstract) QoS specification is taken as required QoS specified at the higher level and the specific supporting (Offered QoS) of a certain implementation platform or a specific resource on the platform that the serves the request it taken as the realization. The MDA approach that we are working with resembles this second aspect.

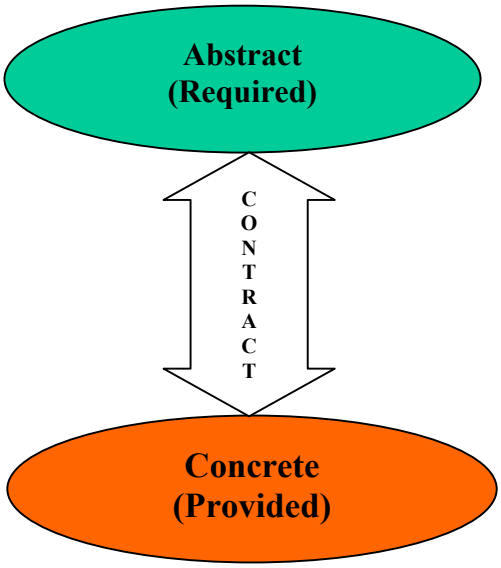

Figure 2-7: Layered architecture of QoS relationship

As discussed above, the notion of QoS (i.e Specification, Mechanism, and Monitoring) is integrated in some networks at the data link level protocols [28,29,31,32].

The two major categories of QoS mechanisms in Link Layer networks are **Reservation** and **Priority**. In reservation, network resources are allocated based on signaled requests originating from applications. Several parameters are used to define the reservation requirement and provision. Signaling messages are used to exchange such parameters. In prioritization, exchanged packets or frames are usually associated with a priority value that defines the handling in relation to other priorities. Sometimes nodes or entire messages can be assigned a permanent priority value that is defined at the setup time of the network.

Several mechanisms for providing QoS exist in both categories. For example Bluetooth and IrDA use the different reservation mechanisms.

2.4 Extensibility of MDA

The MDA key standards (UML, MOF, and XML/XMI) are extensible so that they can be improved (extended) to fulfill the special nature of the embedded world and be implemented in the process of the development of real time and embedded applications. Moreover, the relaxation of the strict constraints in the future embedded applications can lead to an approach that links MDA and embedded system development.

Most extensions are made to the UML that serves as a modeling tool in the MDA.

2.4.1 The UML Profiles

UML is an object-oriented modeling language standardized by the Object Management Group (OMG) mainly for software systems development. It consists of a set of basic building blocks and rules that dictate the use and composition of these building blocks, and common mechanisms that enhance the quality of the UML models [15].

Its rich notation has made UML a popular modeling language in multiple application domains for system documentation and specification, for capturing user requirements and defining initial software architecture.

To make the system specification phase easier and more flexible, UML provides multiple diagrams to model a system from several perspectives or at multiple level of abstraction. Its extensibility feature through standard extension mechanisms: **Stereotypes, Constraints**, and **Tagged Values** are making UML a good modeling tool for other system development domains [16]. The major extension mechanism is stereotype. A stereotype is a kind of model element that extends another model element (its base) and makes the meaning and the usage different than the base. For example modelers in business modeling area distinguish business objects and business processes as special kinds of modeling elements whose usage is distinct in a given development process. These special modeling elements are considered as special kinds of classes having their own attributes and operations. The stereotypes listed in Table 2-3 are example stereotypes with their base model elements in UML.

The extensions to the original UML specification are documented as **UML profiles**. A number of profiles have been proposed for specializing the original UML towards specific domains. Some examples are:

- **The UML Profile for EDOC (Enterprise Distributed Object Computing)** is used to build PIMs of enterprise applications. It defines representations for entities, events, process, relationships, patterns, and Enterprise Collaboration Architecture. As a PIM profile, it needs mappings to platform-specific profiles.
- **The UML Profile for CORBA:** which defines the mapping from a PIM to a CORBA-specific PSM.
- **The UML Profile for EJB:** which defines the mapping from a PIM to Enterprise Java Beans specific PSM.

Real time and embedded systems is one of such domains that is benefiting from such extensions. The general QoS and Fault tolerance specification profile [9] and Schedulability, Performance and Time (SPT) Specification profile [10] have been released by OMG.

2.4.2 The UML Profile for Quality of Service and Fault Tolerance

The QoS profile provides an enabling mechanism to specify and quantifiably define QoS characteristics such as Latency, throughput, response time and availability, in a flexible manner.

The central element of expressing QoS concerns is the notion of ***QoS Characteristics***. A QoS characteristic is a quantifiable property of a model element. It is specified independent of the element it quantifies. It is serving as the constructor of non-functional aspects such as latency, throughput, etc.

QoS characteristics are quantified using ***QoS Dimensions***. Different QoS characteristics may be quantified using different dimensions. For example, **Availability** is quantified either using the "Time To Repair" or the "Time To Failure" dimension. Throughput is usually quantified by a "**rate**" dimension. Finally, maximum, minimum or average dimensions quantify ***Delay*** or ***Latency***.

A ***Direction*** is a term used as an attribute of QoS characteristics that indicates whether large values or small values are better. For example, the direction ***increasing*** indicates that large values are better where as ***decreasing*** indicates that small values are better. Considering throughput, *increasing* is its direction attribute value so that larger throughput values are preferred.

Unit is an attribute of a QoS Characteristics (or Dimension) in which the values are expressed. For example the unit of the dimension Rate can be bps, Bps, etc.

Often QoSCharacteristics may need to be grouped in a ***QoS Category***. For example Performance is a category that includes Throughput, delay, efficiency, and demand [9].

According to [9], the expression that describes the QoS specification is called **QoSConstraint**. It is an expression built to describe limits of one or more QoS Characteristics incorporated in the constraint expression or more formally a Quality of Service expression. There are three types of QoS Constraints. **QoS Required**, **QoS Offered** and **QoS Contract** (**QoS Effective**). The QoS Contract is usually set after the required and the offered QoS are negotiated between a provider and a requester of a service.

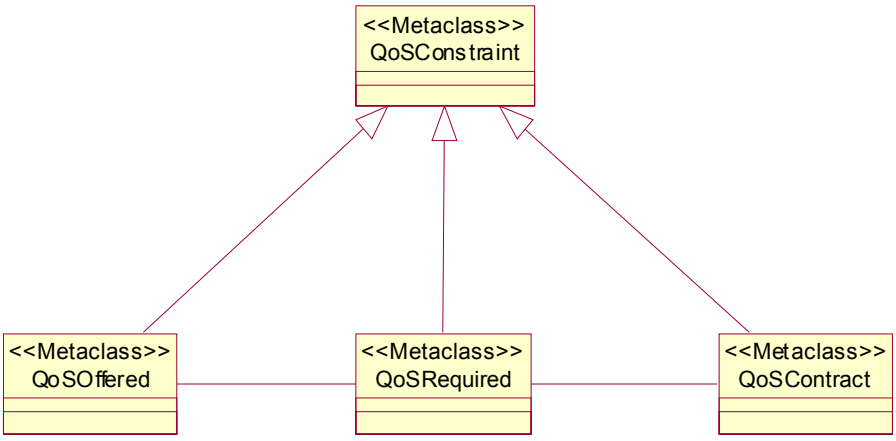

Figure 2-8: The QoS Constraint Diagram [9]

2.4.3 The UML Profile for Schedulabiliy, Performance, and Time

The SPT profile provides a quantifiable notion of Time, Resources and Performance that enables system specifications to include such nonfunctional parameters using scientifically and mathematically derived results. The central element of this profile is the notion of Resource modeling. It introduces a framework named **General Resource Modeling** framework (GRM).

According to this profile, a **Resource** is an entity on an execution platform (software or hardware) that offers one or more services with which a measure of effectiveness or Quality of Service is attached. It classifies resources of an execution platform in to three types namely, processor resource (CPU, Memory), Communication resources (Protocols, Mediums), and Devices (Input/Output).

2.4.4 The Integrated Resource and QoS Profile

For the purpose of our work, we will use the integrated profile of QoS and SPT metamodels discussed above. Their integration will provide the framework for modeling platform elements such as the communications sub-system with formal expressions that can describe both the functional services and the Quality of Service provisions. The artifacts and the constructs defined in the two metamodels will serve as a base for our approach. Both the proposed PIM, PM and the mapping will be defined based on this integrated profile shown in Figure 2-9.

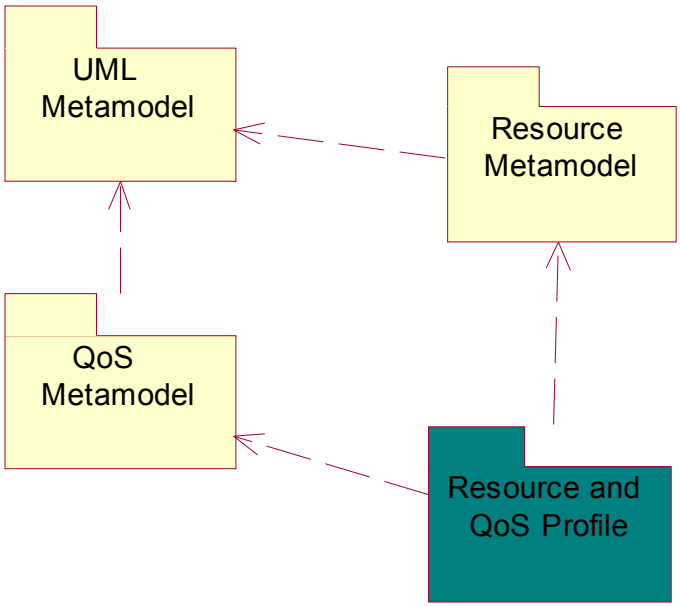

Figure 2-9: The integrated QoS and Resource profile.

A summary of the extensions (stereotypes) introduced within these profiles that we intend to use in this thesis are listed in Table 2-3:

Stereotype	**Base class (UML Core)**	**Comments**
GRMResource	Class, Classifier, ModelElement	Represents a resource
QoSCharactersitic	Class, Classifier	Represents a quantifiable characteristics of QoS associated with a resource
QoSDimension	Feature	Represents a means of quantification
QoSConstraint	ModelElement	A general constraint expression involving one or more QoSCharacteristics
QoSOffered	ModelElement	Extends the QoSConstraint
QoSRequired	ModelElement	Extends the QoSConstraint
QoSCategory	Package	Used to group related QoScharacterisics
QoSValue	InstanceSpecification	Represents a value associated to an instance of a QoS characteristics

Table 2-3: The QoS and Resource profile extensions

2.5 Summary and Conclusions

This chapter has summarized the basic concepts and frameworks that will build the basis of the rest of the chapters and lay the ground to understand the rest of the contents of this document. As discussed, MDA has no firm ground in the domain of pervasive systems and particularly in real-time and embedded systems.

The objective of MDA is separating concerns between generic system models and from specific models and implementations. It is intended in this work to address one of the sources of variability of the system platforms i.e., the communications subsystem that has a very large number of implementation choices.

The focus of this work will be the reservation-based networks whose typical examples are IrDA and Bluetooth, which employ a connection oriented communication link for QoS aware applications.

In chapter 4, detailed analysis of these networks is presented.

CHAPTER 3. Related Works

3.1 MDA for SOC

An MDA approach for System on Chip Design methodology of embedded systems development has been addressed by the work in [35]. This work introduced an MDA design methodology called the "Y" design, which employs co-design architecture for hardware and the corresponding software. This methodology starts by creating a PIM for both the software and the hardware models and then continues by creating an association PIM that is the result of the matching (fusion) of the model elements of the two PIMs. The methodology then continues by transforming the association PIM to a deployment PSM that targets specific programming and execution environments.

This approach requires that both the hardware and software functionalities must be identified for some specific purpose in that there must be a matching between the two model elements. It is more appropriate for systems dedicated to specific tasks such as signal processing so that the functionality can be modeled for both the hardware and the software with the co-design methodology.

However, it does not specify how the QoS issues are dealt with and how the mapping is made. Moreover, it does not consider interconnection between remote entities and how communication protocols and their variability are handled. It takes the hardware architecture as only the machine elements such as buses and chips and not the API level abstractions.

3.2 The Notion Of Abstract Platform for PIM Definition

This work in [36] introduces the notion of abstract platform that can reflect the characteristics of different platforms at a higher abstraction level. An abstract platform represents an ideal abstraction model of infrastructure characteristics. It can represent platform services and other characteristics such as QoS at a higher level relevant for applications. According to this work, platform independence can only be defined after the capabilities of potential target platforms are established. Another observation presented is that different application characteristics or different sets of target platforms generally lead to different types of intermediate models, design structures or patterns, and model transformations.

There are two major requirements of the abstract platform model. First it has to support models at the PIM level, i.e., it presents modeling artifacts that can be used in platform independent models by software developers. Second it has to be mappable towards a set of concrete platforms that are possible targets of the development process. Therefore, a thorough analysis of possible target platform characteristics must be made to define the abstract platform so that there is a balanced gap between the abstract platform and concrete platforms. This will be the modeler's decision.

The main domain used to verify the approach of this work is general systems at the enterprise level. The platforms considered are the component-based platforms such as CORBA and EJB. The modeling elements considered are components and interfaces at the enterprise middleware level.

The critical idea that we have taken from this research is that resolving the two conflicting objectives is very important (i) Platform Independence in simplifying the effort of application designers through abstract platform artifacts and (ii) Mapping simplicity towards concrete platform models. This is a very important concept that should be also applicable in the current and future design approach for embedded systems development. Platform based design is dominant in this domain [16,17]. The gap between the PIM where abstract platform models are used and PSM where concrete platform models are used should be carefully determined so that the two conflicting objectives described above could be balanced. Our defined approach is based on this critical concept, i.e., studying a family of platforms and defining a platform independent (abstract) model with a formal mapping methodology to each target (concrete) platform based on the MDA philosophy. We have tried to define a reasonable gap between the Platform Independent Model and the Concrete Platform Models.

3.3 Network Protocol Modeling with UML

There are two major works done in this aspect that demonstrated approaches for modeling network protocols and their services using the UML language. A first attempt has been made by Sekaran [37] for modeling a data link layer protocol specifically the L2CAP layer of Bluetooth. It describes the steps involved in the conceptualization, analysis and design of the L2CAP (Logical Link Control and Adaptation Protocol) of the Bluetooth architecture. It used UML as the formal modeling language. This work has shown how the static, dynamic and operational aspects of the layer are captured using various UML diagrams. However, the major drawback of this work is that it does not consider the QoS provided at the link layer. It has only considered the functional

services of the layer. Moreover, with the MDA standards it can only be considered as platform modeling (PM) approach for the link layer.

Another similar work is that made by Thramboulidis and Mikiroyannidis [38] for modeling the TCP. It has shown an object-oriented implementation for the TCP layer named OOTCP. The implementation has shown slight difference in performance as compared to the traditional implementation, which is open for improvements. The major contribution is that it has shown an object oriented approach for the analysis, design and implementation of a communication protocol and described the major modeling concepts and model elements that should be used in a UML diagram to describe TCP and its services. It also introduced some extensions to the standard UML intended to define facts and circumstances in protocols such as concurrency and synchronization. However, the QoS issues have not been included in the model.

These two works have shown that object oriented modeling and implementation of communication protocols has a key advantage. Although there is slight performance degradation is expected, the modeling possibility makes the networking and protocols domain to welcome advantages of object-oriented modeling where extensibility, reuse and interoperability could be practiced.

In our approach, we have taken the basic modeling elements identified in these works (such as Connection, Message, and Event) and extended them with the associated quality of service provisions of the networks. The QoS modeling concepts defined in UML Resource and QoS profiles are used to propose a formal modeling framework for the domain of the networks considered in this work.

3.4 The Platform Modeling Approach

[15] Presents an approach to model network platforms by introducing UML extensions. It describes that the basic elements of a network platform that should be modeled are the **Protocol (stack),** the **Physical medium** and their services, the **Ports** and the **Nodes**. It has shown that the different protocol layers can be taken as different abstraction levels of the communications platform. It also introduced a modeling approach called **UML Platform** that can be used to describe embedded system platforms including the communication subsystem at different abstraction levels. This approach has suggested several extensions to the standard UML in addition to those introduced in the UML profile for Shedulability, Perfomance, and Time [10]. It has shown a demonstration of the approach using the intercom voice communication system case study, a single cell in which a group of users can make voice communications with different QoS

requirements. It has shown how that the UML extensions can be used to represent platform concepts in a real problem in the test case.

However, it does not include how the actual mapping is made between the services of different abstraction levels, such as mapping requirements of upper layer or application requests with provisions of lower layers specially the QoS aspects.

3.5 Quality of Service modeling approaches

Modeling Quality of Service, a general term that is used to describe non-functional aspects of systems, is among the important issues addressed by different research works recently. Several approaches for modeling QoS have been proposed. Many of these researches consider the general descriptions for QoS. And most of them are component based targeting at extending the modeling languages used to describe general software systems at the enterprise level such as UML or introduce a specific QoS modeling language [39,40,41]. They do not address specific domains that are closely associated with QoS such as networks and embedded systems.

Weis Torben [41] has shown how QoS issues can be incorporated at the component based enterprise middleware technologies such as CORBA, .NET, and Web Services. It also introduced a QoS enabled MDA approach that includes PIM, PSM and transformation technique called Kafka. However, it considers the contract between a client component and a server component in a distributed system, which is horizontal relationship between them. It takes system components at the same abstraction level as service client and provider. It has not considered the effects of the communication networks and the underlying infrastructure between them.

In [42], Aagedal presents the concept of orthogonal separation between the QoS specification and the functionality specification of a system. The major reasons for separately dealing with QoS issues are:

i. The QoS of a QoS component instance is dependent on its environment, a QoS component can be a software or hardware element (resource) that gives one or more services and can support certain QoS constraints.

ii. There may be multiple QoS specifications for the component, and

iii. The functional specifications of the component are independent of the particular QoS specification that will be enforced at any point in time.

Furthermore, it has shown how to link QoS aspect models with functional elements of models called computational elements such as Actor, Component, Interface, Node, Object, Subsystem,

Use case, and Use case instance. But it does not use the MDA concepts such as Platform Independent Modeling, Platform Specific Modeling and Transformations.

3.6 Summary and Conclusions

The above works have shown different aspects of modeling functional and nonfunctional (QoS) aspects of systems at different levels and domains. As MDA is an evolving methodology, only few works have been done so far along its track. There are no generally accepted approaches that have incorporated both functional and QoS aspects in the MDA modeling levels (PIM and PSM) and transformations. Most works are concentrating on either PIM modeling or PSM modeling alone without showing the actual mapping or transformation strategy. Moreover, most of the approaches that include transformations and mappings consider mainly the functional and structural aspects of models. Mapping QoS aspects at different abstraction levels is the least considered area, which is a crucial aspect for the adaptation of MDA towards the embedded systems domain.

Embedded systems are complex and different from other software development domains because they are closer to architectural rather than abstract application models. Most of the embedded platform services are difficult to model at an abstract level specially the QoS aspects. Therefore many of the recent works largely concentrate on extending the modeling languages typically the UML towards the domain of embedded systems and QoS.

The works lack a complete MDA approach that can enhance systems and application development in this resource and QoS aware domain. Further works that present PIM, PM, and the mapping approaches are required. No work has been done that has considered the variability of the networks used by embedded systems in terms of functional and non-functional properties with the MDA concepts and standards.

However, we have observed that the works discussed in this and the previous chapter lay the ground for the complete MDA approach. From the works we have learned that networks are part of platform resources, which provide interconnection services with quantifiable properties. A modeling framework and metamodel standard is presented in the UML QoS and Resource modeling profiles. The possibility of Object Oriented modeling and implementation of communication protocols (services) has been proved promising by some works. Platform abstraction is introduced as key methodology to address extensibility. The SoC domain also has shown a proof of the applicability of model driven methods in the embedded systems domain.

Using these works as a background support, we propose a formal MDA approach for the development of embeddable communicable applications. Focusing on the network subsystem of embedded platforms, we introduce a QoS aware model driven approach that will help the development of such applications to be network independent with strategic mapping towards a specific network. The approach can be used to model the communication interface at network independent level (PIM), network level (PM) and the mapping between the two.

CHAPTER 4. The Proposed MDA Approach

4.1 Overview of The Approach

In enterprise systems, a design of an application or software system will have two distinct design models namely the PIM and PSM. The PSM is a new production of a mapping or transformation performed on the PIM. Here, the Platform Specific Model is a new model produced to reflect the characteristics of the software system with platform specific artifacts. It is this PSM that is will be implemented in that specific platform.

The PIM level of this domain only shows components, packages and classes with their structural and functional characteristics, which is independent of technology artifacts used with CORBA, EJB, or DCOM, which are possible implementation platforms. The transformation creates a model (PSM) of the initial PIM according to its structural and functional elements and their characteristics for either of those possible target platforms. A generic *Interface* element in the PIM will be modeled as a *Stub* and *Skeleton* objects in CORBA and EJB, and as *proxy* and *Stub* objects in DCOM. Moreover, the way interfaces and objects are associated differs in each of these platforms. Hence, the same PIM model will be first transformed to either PSM, which is yet a design model of the system. The PSM will then be ready for implementation possibly through code generation support from MDA enabled tools.

Such an MDA process is not generally suitable in the current and future embeddable communicable applications. The existence of the QoS issue makes most of the system models to focus on the runtime application behavior and environment characteristics, which is generally difficult to determine at design time [21]. Most of the applications must first identify their environment specially concerning QoS, which can be identified only at run time. Because of this nature, the application model of embedded systems in general includes the model of the platform on which they are going to execute.

This general scenario indicates that the application model and the platform model are closely related so that platform models in the embedded system development methodology greatly influence application models. The major concern is how to model the applications in order to use

specific environments efficiently. Hence the model of the applications usually follows the model of the execution environment or is made along with the design of that specific environment (Co-design).

All these facts confirm that it is the platform, which has a major part in the embedded application design instead of the intended application since the entire functionality of the application will be limited by the supporting capability of the platform [46]. In particular, the QoS required by the application must be supported by the target platform, which influences the extensibility, reusability and portability (platform retargeting capability) of the application by focusing only on the currently intended target platform and its resources.

Hence, unlike enterprise systems, the MDA approach for embedded systems in general should be based on the platforms and their abstraction instead of application models and their refinement. The notion of "Abstract platform" [36] tailored with the MDA methodology will leverage the current challenges and visions in the embeddable communicable applications development. If target platform models can be represented abstractly and standard mappings and transformations are defined towards lower layer refinements and implementations, the design of embedded systems will be simplified and the implementation dimension of applications will be widened.

Therefore, what we propose is a model driven platform based (resource oriented) approach for embeddable communicable applications that use different platforms composed of the three standard classified elements, namely OS (RTOS), Communication, and Device Driver subsystems. As described earlier, we focus on the local (short-range) view of pervasive systems that employ small devices and short-range networks where many choices exist. The variety is incomparable with the verity that exists in enterprise and desktop system platforms.

We propose that the best way of adapting the MDA approach towards embeddable communicable application development is introducing the MDA concepts on the platforms instead of the application models. This way, the PIM of the platforms will be an abstract model that can be used within the model of the applications. Upon implementation the abstract platform will be mapped with a specific platform through a mapping layer.

The goal is that applications will be implemented with the abstract platform specification on the devices. For a specific target platform, an appropriate mapping layer towards that target will be implemented on the same device. The applications model is still independent of the target network. Whenever a new platform is targeted, only the mapping layer is modified so that the application still interacts with the mapping layer, which links it with the new target platform.

In this case, there is no distinct instance of a PSM. The PSM in our case is the combination of the Platform Model instance and the Mapping Layer model instance targeting it. This is because the mapping layer is modified for the specific platform it is targeting.

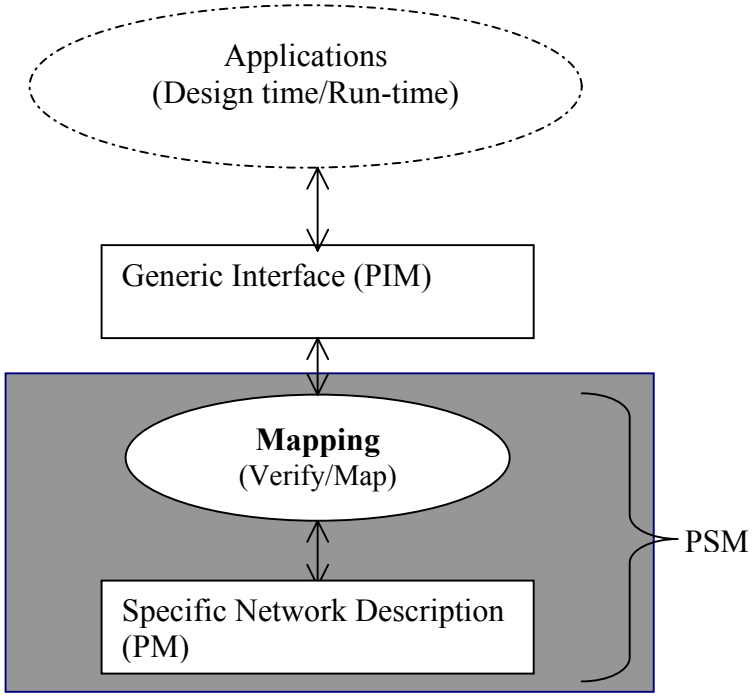

Figure 4-1: A conceptual model of the proposed approach

4.1.1 Assumptions

- We will be dealing with only the communication part of the systems (applications).
- The overhead imposed by the Operating System and Device Driver subsystems of the platforms that are not directly related to the communication subsystem are not considered.

4.1.2 The Used Metamodel (The UML Profile)

As discussed in chapter 2, we use the integrated UML profile of Resource and QoS metamodels as a basis for modeling both the PIM and PM and Mapping of the networks introduced in section 2.4.4. The framework is shown below:

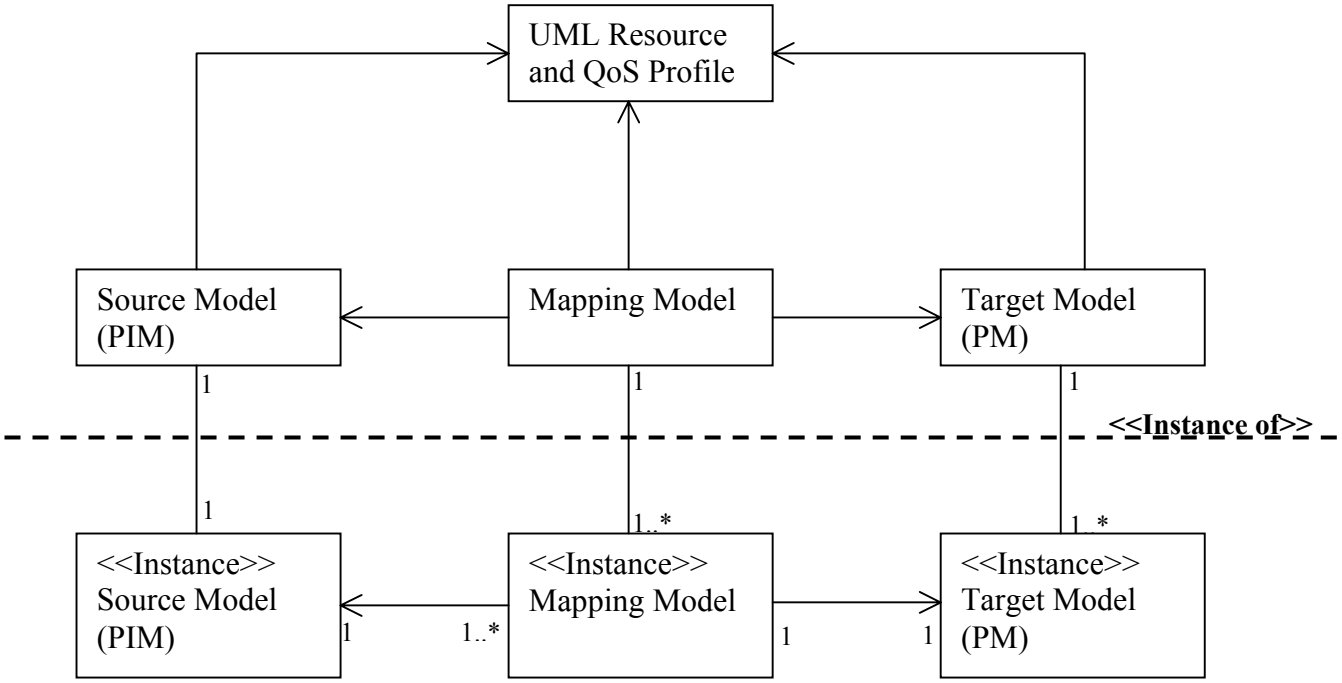

Figure 4-2: The QoS and Resource profile to be used as a framework for in this thesis

As shown in the diagram, there will be only one model and instance of the PIM where as there will be a number of Platform Model instances and an equivalent number of the mapping layers targeting each of them.

4.2 Analysis and Modeling of the platforms

Previously, we categorized and gave an overview of the networks in the embedded domain. This section will present a detailed analysis of the relevant layers and services of the reservation based category networks and their services. Two typical examples of the networks that employ the connection oriented and reservation based QoS mechanism namely Bluetooth and IrDA will be investigated.

4.2.1 Bluetooth

According the Bluetooth specification [29], the Logical Link and Adaptation Protocol (L2CAP) provides (represents) the Bluetooth link layer services. All application level services are specified as profiles that must use the services of the L2CAP layer. Example profiles are LAN access profile and Object exchange profile.

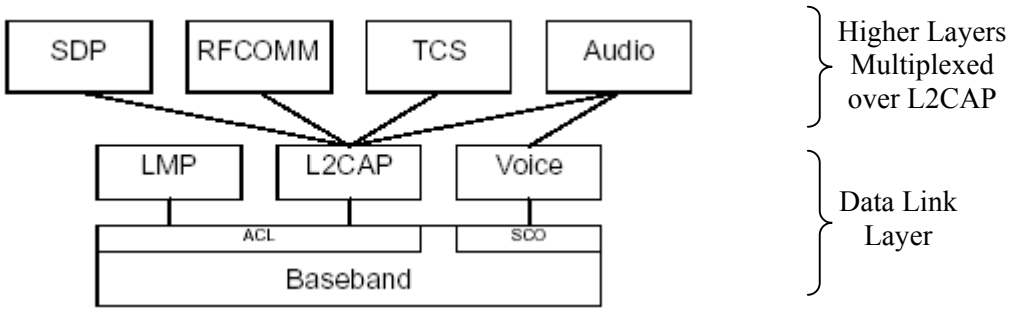

Figure 4-3: The architectural alignment of L2CAP in the Bluetooth protocol layers

The SDP(Service Discovery Protocol) is used by the applications to discover which services are available and to find the characteristics of those available services. Clients use it to search for the needed services based on specific attributes of those services. The function of Telephony Control Service (TCS) is to provide call control using signaling for the establishment and release of speech and data calls between two Bluetooth devices. RFCOMM emulates serial communication for some legacy applications using the RS232 serial port [29,51].

How the L2CAP works

L2CAP works with the concept of channels. It is the sole user of the ACL physical link created by the Baseband layer. There can be only one ACL link between two Bluetooth devices. For each application or client service of the L2CAP layer a different L2CAP channel is created on top of this link.

After device discovery, the Bluetooth Baseband layer creates a single ACL link between the two devices. The other type of Baseband link, SCO, is created on demand when there is as need for voice communication and is not related with L2CAP. L2CAP is the sole user of the ACL link. It is non-transparent to upper layers that use the ACL link.

The general functional requirements of this layer are:

- **Segmentation and Reassembly**: Since the connection channel between two devices will be configured for a negotiated size (Maximum Transmission Unit) between the peer ends, segmentation and reassembly of large sizes are performed by this layer.
- **Protocol Multiplexing**: many applications residing on a device may create their own channels on the L2CAP layer. Since all should use a single ACL link between the two nodes, multiplexing is the task of this layer.
- **Quality of Service**: Every channel is created with QoS negotiation between the two peer entities of L2CAP. Non-QoS aware applications may choose to use the defaults

concerning QoS. The channel is configured for a flow specification in both directions (since Bluetooth supports full duplex channels) with different levels of service guarantees.

Interaction of L2CAP with layers (Its clients and Lower layers)

The L2CAP layer interacts with higher layer protocols through several service primitives. The non-data messages are called signaling commands. Messages that come from the remote devices to L2CAP layer are passed through lower layers as events. Such messages are referred as Confirmations and Indications. On the other hand messages that come from upper layer to lower layers are referred as Requests and Responses. Each of the message exchanges has defined primitives.

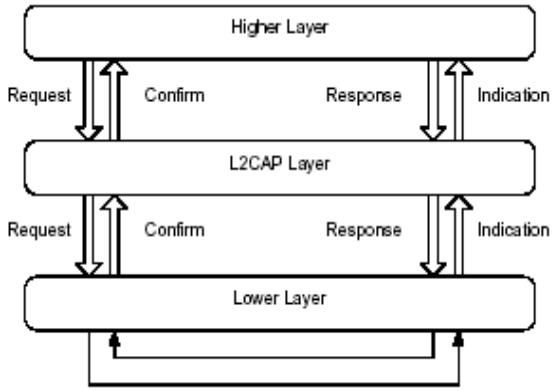

Figure 4-4 Interaction between Bluetooth layers.

4.2.1.1 General Procedure for channel creation and configuration

The general procedure that describes the steps needed to create an L2CAP channel is stated below:

- The initiator end requests for a channel to be created between the devices
  - If the channel is created it will have two end points called channel end points (CIDs) on the pear entities. The initiator will receive a positive confirmation message (event).
- The initiator again requests for the configuration of the created channel by specifying its preferences (the parameters may be requested by an upper layer the client)
  - Since Bluetooth supports full duplex channels, the configuration is made for both directions.
- The data transmission begins based on the negotiated settings.
- Finally a disconnection procedure will be initiated in the same manner.

The LMP is responsible to control the transmission process according to the negotiated settings.

The general interaction between layers around L2CAP is depicted below in Figure 2

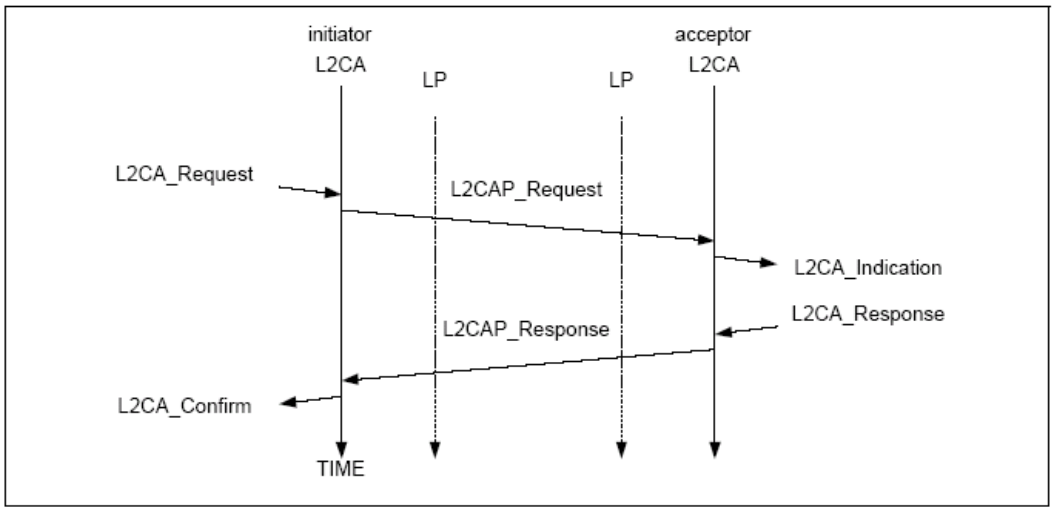

Figure 4-5: The general operational sequence diagram of L2CAP

The state chart showing the states that an L2CAP channel undergoes is shown below. There are three basic states that a channel undergoes from start to termination of the communication process.

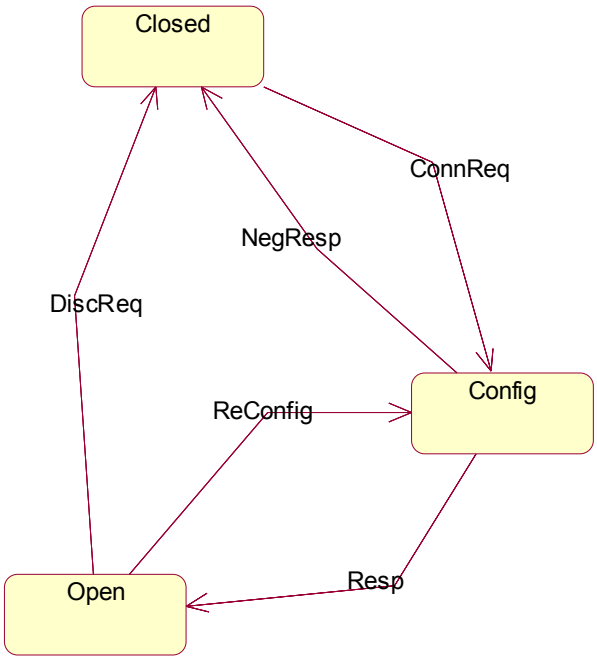

Figure 4-6: The states of an L2CAP Channel

4.2.1.2 Channel configuration parameters in L2CAP

Every channel created at the L2CAP layer is configured for QoS using a channel configuration primitive. The channel configuration parameters are listed below:

Maximum Transmission Unit (MTU)

This parameter specifies the maximum size of a packet that the initiator is capable of accepting. Its minimum value is 48 Bytes and the default value is 672 Bytes.

Flush Time out (FlushTO)

This option specifies the amount of time the originator of a packet attempts to successfully transmit it before flushing it. It is specified in milliseconds.

Quality of Service

This is a flow specification (OutFlow) that incorporates several sub parameters described below. It is a token bucket based specification.

Parameter	Values	Description
Service Type	No traffic Best effort Guaranteed	Defines service level
Token Rate	Rate in bps	A Bluetooth user may send data at this continuous rate
Token Bucket Size	Size in Bytes	Defines the burst size
Peak bandwidth	In bps	Maximum data rate for a Bluetooth user
Latency	Latency in μs	Maximum delay for a bit through a Bluetooth terminal. The delay ending is when the bit is first time transmitted by the radio.

Table 4-1: The Bluetooth L2CAP layer flow specification parameters

The values of the Token Rate, Token Bucket, and Latency depend on the service type. If Best Effort, the request may not be enforced. If Guaranteed, then a maximum possible value at the time of request will be enforced and contracted. The details are discussed in Annex A.

4.2.1.3 The Bluetooth Link Layer Platform Model (PM)

The following diagram shows a general model of the Bluetooth of the Bluetooth platform as abstracted at the L2CAP layer. The platform model of Bluetooth consists of the Interface, Functional elements, and Non functional QoS elements. These elements are used to describe the basic functional or operational characteristics of the Bluetooth L2CAP layer. They are identified and listed below:

The functional services are:

Connection Setup

Connection Configuration/Reconfiguration

Connection Termination

Message Transmission

Data (Send, Receive)

Event Signaling

The objects (entities) identified are:

Channel/Connection: this refers to the channel identified with two endpoints on peers.

Event: every message exchange is produced as an event, which invokes a corresponding operation

QoS_Spec: this represents the QoS constraint (**Offered QoS**) of the Bluetooth L2CAP layer.

Service interface: this represents the L2CAP service entity through which clients interact with the layer.

Classes will be used to represent these four entities. Using the terminology defined the UML profiles, Channel/Connection is considered as a Resource and QoS_Spec is a QoSConstranit. The other elements are modeled using the standard UML concepts.

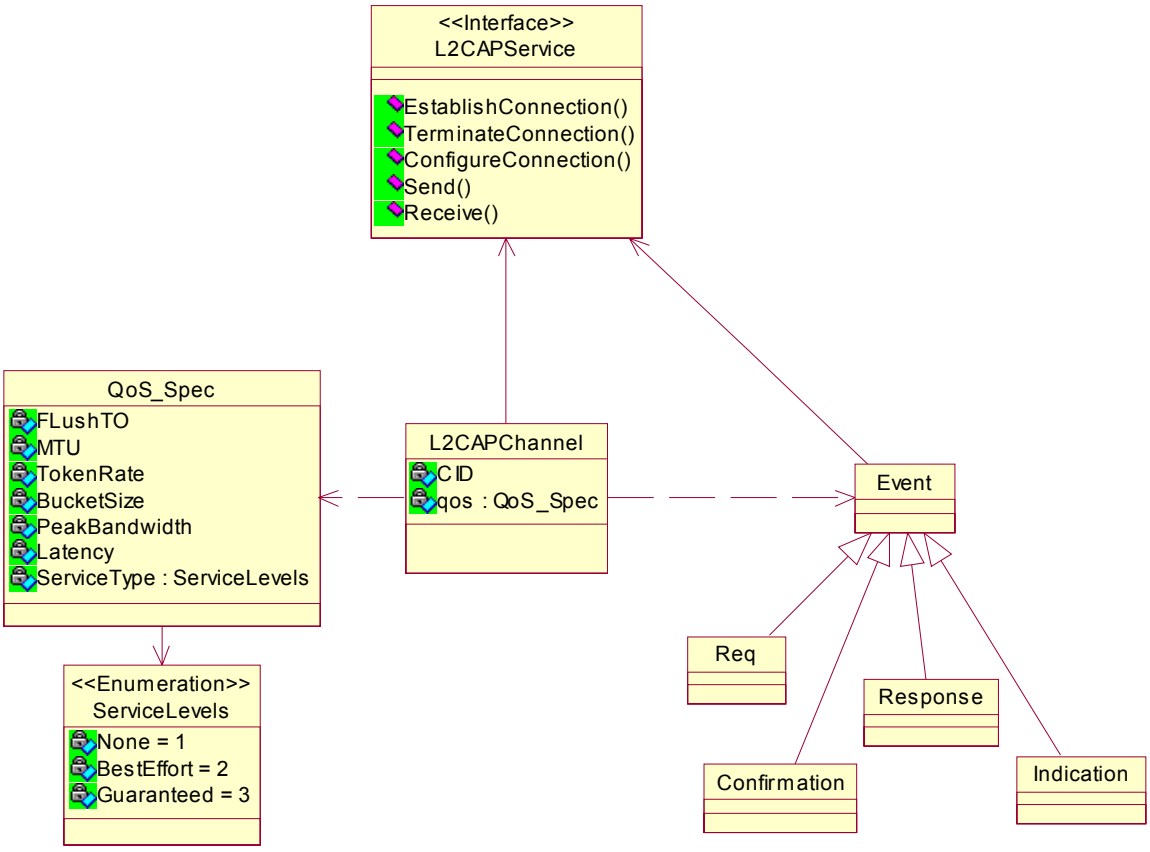

Figure 4-7: An L2CAP platform model Static model using UML

According to the terminology defined in [9], the parameters used to define the QoSSpec for Bluetooth are each QoSCharacteristics having their own quantifying dimension and values. They are: **MTU**, **FlushTO**, **TokenBuketRate**, **TokenBucketSize**, **PeakRate**, and **Latency**.

4.2.2 IrDA

The lowest non-transparent layer of IrDA is the IrDA Link Management Protocol (IrLMP). This layer presents the link layer services of the IrDA network for higher layer users of the network [56][57].

The actual channel or link will be created at the IrLAP layer but this layer is totally abstracted by the IrLMP layer. The major purpose of IrLMP is to provide a multiplexed service over the single link created by the IrLAP layer. IrLAP resembles the Basband layer and IrLMP resembles the L2CAP layer of Bluetooth respectively.

4.2.2.1 Link layer Services

There are two major services provided by the LMP layer of IrDA [57]. A brief summary of them is given below.

Information Access Service (LM-IAS)

This entity stores an information base of services provided by applications on a device to be discovered by other devices upon request. A number of objects store this information, which can be viewed as an external conceptual view of the services through the object attributes and operations.

Link Management Multiplexer Service

This service interface provides both the link management and the LSAPs (Link Service Access Points). After discovery by LM-AS the LSAP points will be created at the LM-MUX interface to enable the peer applications to interact with each other.

The creation of the LSAPs follows the creation of a corresponding IrLAP link and the discovery of the required application service by the IAS. LSAP creates several link end points for many applications to get a multiplexed access over a single IrLAP link.

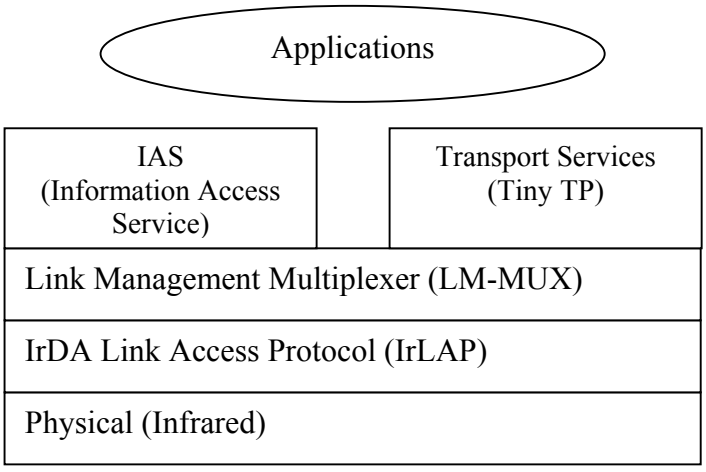

Figure 4-8: The IrDA architecture

The channel creation and configuration process is similar but slightly different from Bluetooth L2CAP. The IrLMP Multiplexer (LM-MUX) creates channels with endpoints grouped at an interface element (LSAP). The connection and configuration process is made at the same time using a single primitive (LM_Connect). The primitive initiation and transmission events are similar to that of L2CAP.

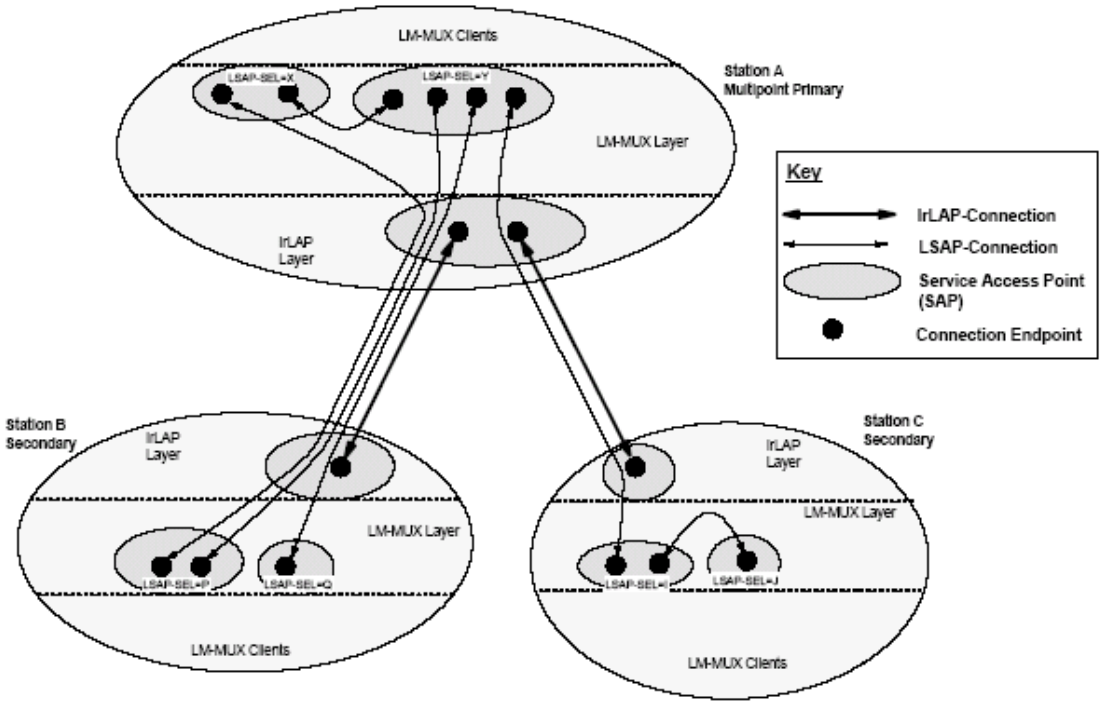

Figure 4-9: General relationship between the IrLAP links and LSAPs or LM-MUX [57]

The primary purpose of the LM-MUX is to provide connection-orientated data transfer services between multiple LM-MUX clients (e.g., transport entities or directly bound attached applications). Peer LM-MUX clients are connected by an LSAP-Connection.

Figure 4-9 illustrates the relationships between Stations[1], LSAPs, LSAP-Connections, LSAP Connection Endpoints, IrLAP-Connections and IrLAP-Connection Endpoints. LM-MUX provides connection-orientated data transfer services between multiple LM-MUX clients (e.g., transport entities or directly bound attached applications), Peer LM-MUX clients are connected by an LSAP-Connection.

4.2.2.2 General Procedure for channel creation and configuration

The common procedure that describes the steps needed to create an IrDA link is stated below:

- An application requests for a communication with a peer entity on an other device
- The LMP requests the IrLAP layer for connection
- The IrLAP creates and configures the connection
- The IrLAP returns a confirmation for the requesting IrLMP entity (requesting end)
- The IrLMP layer sends a connection with a peer LSAP application end point

[1] Three stations are used to show the general scenario. Currently only two stations can be linked by IrLAP [57].

- The peer application will be instantiated and responds
- The response will be propagated back to the initiator
- Communication then begins

All the QoS parameters are incorporated with the IrLMP and IrLAP interactions while setting up the link.

4.2.2.3 The connection creation and configuration primitive

The IrLMP layer provides a single primitive that is invoked to create and configure a channel to be used for information exchange by its clients. The primitive is called **LM_Connect** with four versions that represent the signaling steps between the peer entities.

The LM_Connect primitive

Once the LSAP for a client entity on a remote device has been identified, the LSAP for the local application (transport) entity and the remote entity must be bound (linked) for data to be sent. There can be at most one LSAP-connection between any given pair of LSAP entities linked using the following four massage sequences. The **request**, **response**, **confirms**, and **indication** primitives have the same semantics and purpose as those in the Bluetooth L2CAP.

LM_Connect.Request(Called LSAP, Requested QoS, Client Data)
LM_Connect.Indication(Calling LSAP, Resultant QoS, Client Data)
LM_Connect.Response(Calling LSAP, Client Data)
LM_Connect.Confirm(Called LSAP, Resultant QoS, Client Data)

Parameters:

LSAP: A reference to an LSAP (typically an LSAP-ID of a remote end point)

QoS: Quality of Service parameters. The parameters that can be changed by a request from the IrLMP layer are listed below.

- Baud Rate
- Max. Turn Around Time
- Data Size
- Window Size
- Disconnect Threshold
- Minimum Turn Around Time

- Details of the QoS parameters are discussed in Annex B.

Client Data: optional client information sent with the request.

The sole service provider at the Link layer of IrDA, IrLMP by itself is subject to the negotiated parameters at the IrLAP layer even if it can initiate the connection setup procedure following application or higher layer requests.

A LSAP service user may request a Quality of Service (QoS) for the IrLAP link. Then an attempt to provide the requested QoS will be made. The connection will not be rejected simply because the QoS could not be met. If the connection succeeds, the actual QoS parameters will be returned. If the actual QoS is not sufficient, it is up to the LSAP Service User to disconnect.

The diagram below shows the general LSAP connection states

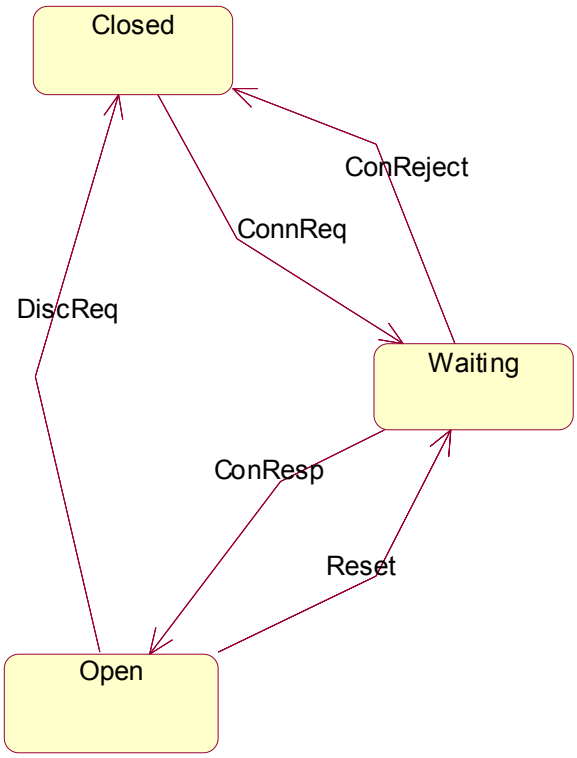

Figure 4-10: State chart showing the states of an LSAP link

4.2.2.4 The IrDA link layer platform model (PM)

The following diagram shows a general model of the IrDA platform as abstracted at the LM_MUX layer. Similar to the Bluetooth L2CAP platform model, this platform model of IrDA consists of the Interface, Functional elements, and Non functional QoS elements. These elements are used to describe the basic functional or operational characteristics of the LM_MUX layer. They are identified and listed below:

The functional services are:

Connection Setup

Connection Configuration/Reset

Connection Termination

Message Transmission

Data Send and Receive

Event Signaling

The objects (entities) identified are:

Channel/Connection: this refers to the LSAP with endpoints at peer LM_MUX layers.

Event: represents the means of message interaction between objects in the system. Every message is produced as an event to invoke a corresponding operation.

QoS_Spec: Represents the QoS Constraint (Offered QoS) of a IrDA Link layer.

Service Interface (LMService): this is the interface with the Link layer interacts with its clients

Similar to Blutetooth, classes will be used to represent these four entities. Using the terminology defined the UML profiles, Channel/Connection is considered as a Resource and QoS_Spec is a QoSConstranit. The other elements are modeled using the standard UML concepts.

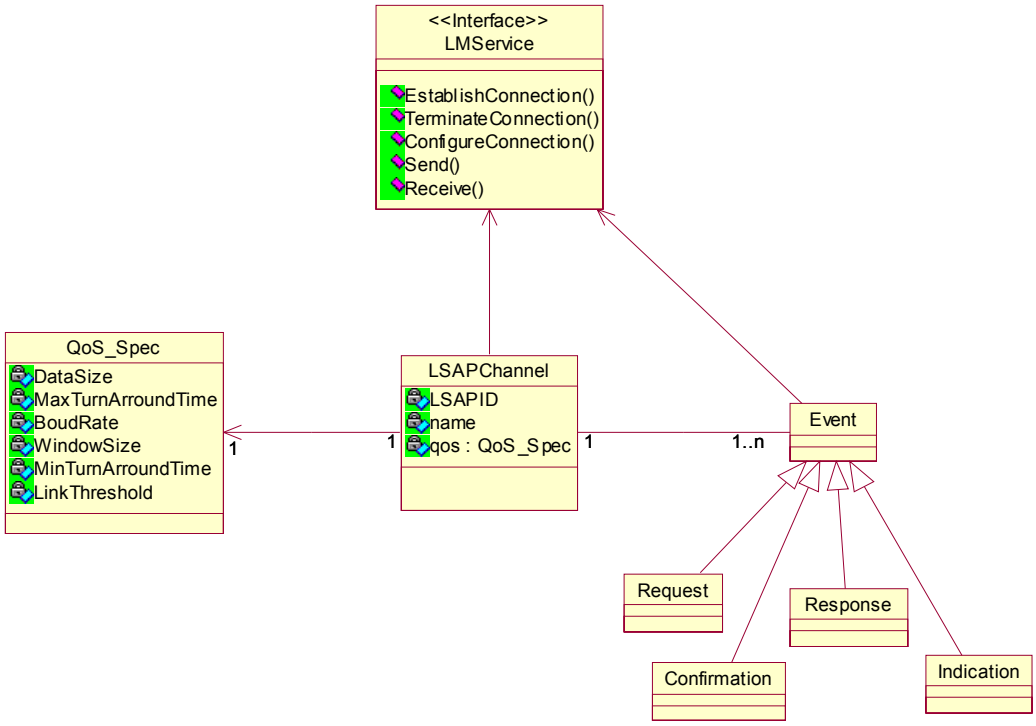

Figure 4-11: The IrLMP Platform Model

Additionally, all the QoS_Spec parameters have enumerated values. All of them will have an enumerated data type definition given in Annex B. No other values should be used.

According to the terminology defined in [9], the parameters used to define the QoS_Spec in IrDA are each QoSCharacteristics having their own quantifying dimension and values. They are: **Maximum Turn Around Time**, **Link Threshold Time**, **Baud Rate**, **Data Size**, **Window Size**, **Minimum Turn Around Time**.

In summary, we can see that the model elements of both IrDA and Bluetooth link layer services are closely similar. They both have the Connection, Event, and ServiceInterface and QoS_Spec objects. The major difference between the two models is the difference in the definition of the QoS_Spec model element.

A general modeling framework for reservation based and connection oriented platforms modeling can include the following concepts as Metamodel elements, which will have their own definition for each network in the considered reservation based domain.

The major platform metamodel framework can be assumed to include:

- Channel
- Event

- QoS_Spec
- QoSCharacteristics
- QoSDimension
- QoSValue
- ServiceLevel
- ServiceInterface

All network specific platform models will be specific instances of this metamodel framework.

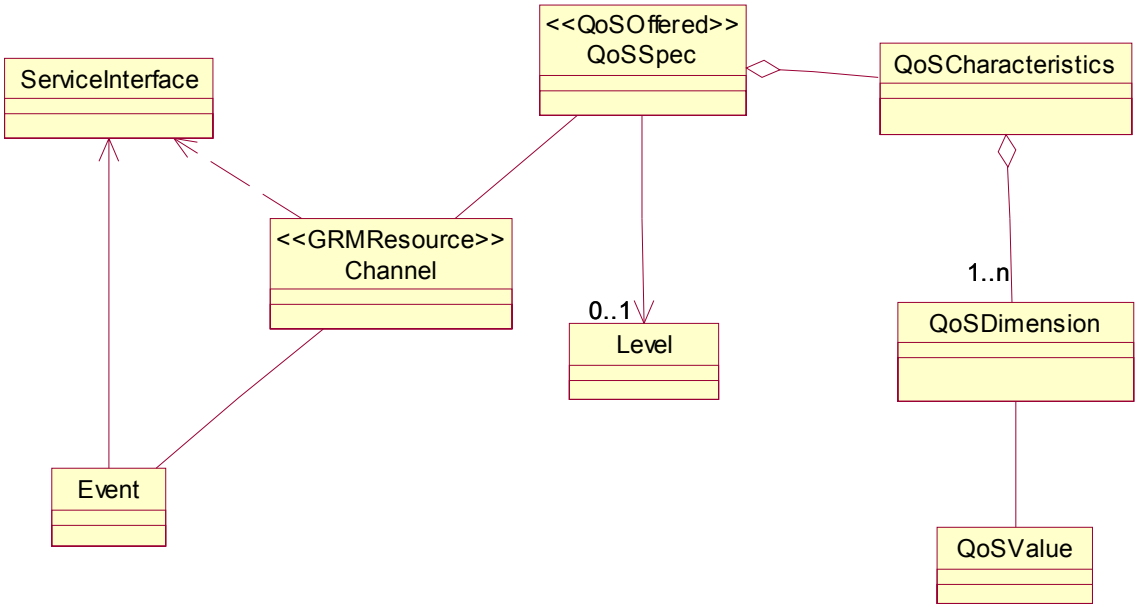

Figure 4-12: The Platform Framework Metamodel

4.3 The Platform Independent Model (PIM)

As described in section 4.1, the main objective of the PIM is to represent the services of the domain of networks in a generic manner so that application design and implementation can be independent of the underling network. Although it is an implementation decision, the model can be incorporated with in application design and implementation or can be implemented as an abstract layer on the devices. A more precise name for this model could be Network Independent Model (NIM).

At this level specific network level artifacts should be abstracted out so that the following two requirements of a PIM should be fairly satisfied.

- Independence of platform artifacts
- Easy mappablity towards specific platform model

With the MDA standard, the PIM should be semantically similar to the platform models. Hence, it has to reflect the connection oriented and the reservation based nature of the networks. The applications implemented in this network domain are aware of the reservation based and connection oriented nature of the networks. But their design and implementation will be independent of a specific network interface. The applications are implemented on the devices and objects with the knowledge of the platform independent service interface. It will be the duty of the mapping layer to transform (map) this interface and QoS specification to the underlying network. Thus, producing a separation of concerns between the applications and the platforms. Therefore the PIM concepts are based on the platform specific characteristics.

The essential model elements that the PIM must include in an abstract manner are:

- **SeriviceInterface**: this is an abstract entity that represents the service interface where clients interact with the network. This specifically abstracts the interface of the networks. It serves as a generic, abstract, and high-level Service Interface for application modelers.
- **Channel**: is the channel with which the QoS request is associated. It is an abstract representation of the connection concept used in each network. The properties are based on abstract definitions.
- **Flow_Spec**: this is the specification of the QoS as a generic reservation request specification similar to the flow specification proposed in [45], which has been enhanced in [21] and further by IETF for use in Internet reservation services [49]. It is believed in this work that it can be mapped or transformed to a number of connection oriented and reservation based networks. The specification does not explicitly define latency. Therefore, we have added a latency parameter in the flow specification.
- **Event**: represents all messages used as notifications for negotiation, and message sending and receiving. This is also an abstract representation of the event model element in the platform models.

For the QoS specification at the PIM level, the flow-based approach is selected for a number of reasons discussed below.

First, it is a closer approach to networks and in particular it is more appropriate for the connection oriented and reservation based networks considered in this work. Initially it was proposed for the Internet community. It is also a widely used model to quantitatively specify application requirements on a network [52].

Second, it is more declarative than showing more technical details, which makes it appropriate for a PIM level specification according to the MDA standard [2,3].

Third, its specification does not target a specific network protocol and reservation mechanism [33,45]. This opens the possibility of mapping to many different specific implementations.

Fourth, we believe that it is the most appropriate specification that can satisfy both requirements of a PIM stated above. We argue that this type of PIM specification can be transformed to Bluetooth, IrDA and other reservation based networks.

[45] presented an abstract network element service specification framework especially for flow-based networks which is summarized below:

> *The definition of a service states what is required from any network element (a router, switch, subnet, or the network subsystem of a host or operating system). The service definition also specifies parameters used to invoke the service, the relationship between those parameters and the service delivered. Each service definition also specifies the interface between that service and the environment. This includes the parameters needed to invoke the service, informational parameters which the service must make available for use by setup and management mechanisms, and information, which should be carried between end-nodes and network elements by those mechanisms in order to achieve the desired end-to-end behavior.*
>
> *However, a service definition does not describe the specific protocols or mechanisms used to establish the required state in the network elements for flows that use the described service.*

Other specifications that can be used for reservation-based network for example INSIGNIA [55] and FQMM [54] are initially based on this generic flow specification adapted towards mobile ad-hoc networks. Moreover, they focus on the routing service on multi-node wireless networks with specific implementation.

4.3.1 Data Structure of Flow Specification

Flow Specification has the following parameters [33,45,49]:

Maximum Transmission Unit (MTU)

Specifies the maximum number of bytes in the largest possible packet to be transmitted over the flow can have.

Token Bucket Rate

The token bucket rate is one of three fields used to define how traffic will be injected into the network by the sending application. The token rate is the rate at which tokens (credits) are placed into an imaginary token bucket. For each flow, a separate bucket is maintained. To send a packet over the flow, an application must remove a number of credits equal to the size of the packet from the token bucket. If there are not enough credits, the application must wait until enough credits accumulate in the bucket.

Token Bucket Size

The token bucket size controls the maximum amount of data that the flow can send at the peak rate. More formally, if the token bucket size is B, and the token bucket rate is r, over any arbitrarily chosen interval T in the life of the flow, the amount of data that the flow sends cannot have exceeded **B + r *T** units.

The token bucket is filled at the token bucket rate. The bucket size limits how many credits the flow may store. When the bucket is full, new credits are discarded. It must be greater than or equal to the MTU size.

Maximum Transmission Rate (Peak Rate)

This parameter limits the maximum transmission rate limits how fast packets may be sent from the host in cases of bursty traffic produced by the sending application. It limits the transmission rate bounds by defining how fast successive packets may be placed on the network.

Therefore, if the maximum transmission unit is M, and the maximum transmission rate is p, over any arbitrarily chosen interval T in the life of the flow, the amount of data that the flow sends cannot have exceeded **M + p *T** units.

Service Level (Guarantee Level)

This parameter has 3 values that indicate the level of guarantee associated with the flow specification. Reservation depends also on this value. The possible values are Guaranteed, Controlled, and Best Effort.

In addition, the flow specification will have the following additional specifications for delay (latency).

Latency

The time it takes for a packet from its delivery to the link layer at the sender side to its delivery from the link layer of the receiver.

4.3.2 The Platform Independent Model Diagram

The platform Independent Model is given in Figure 4-12.

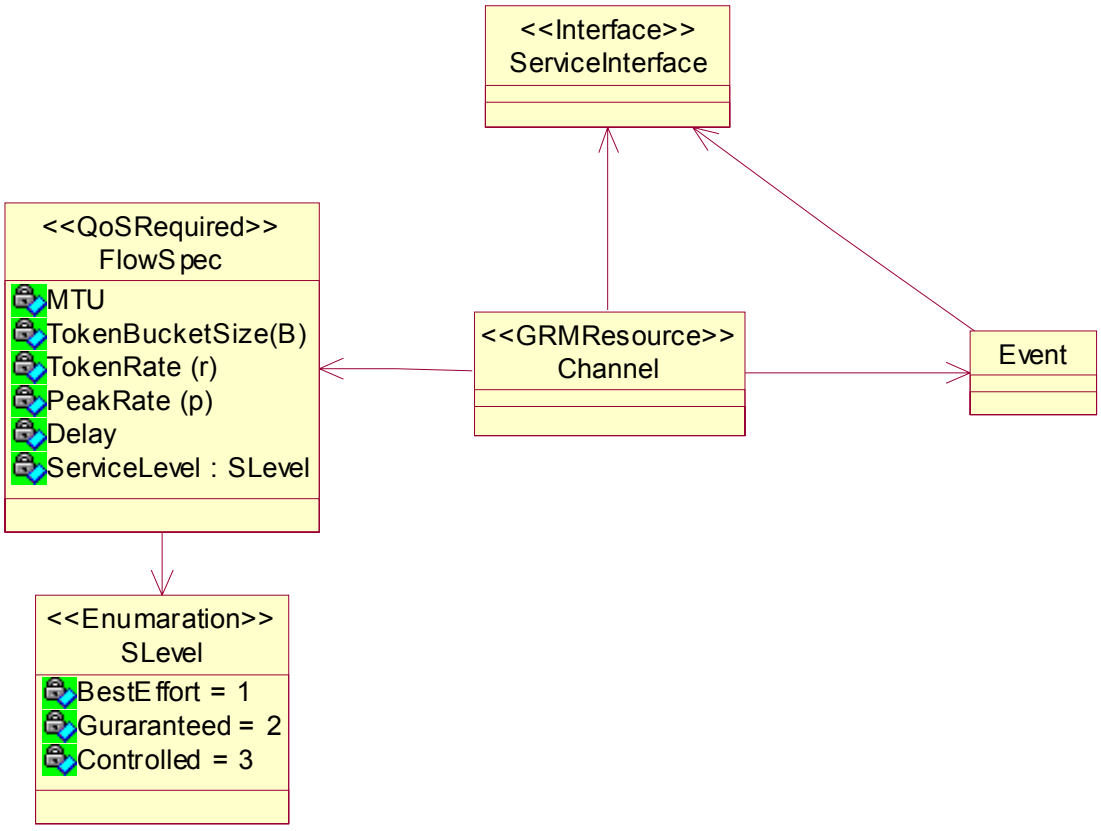

Figure 4-13: The proposed PIM representing the flow based connection oriented networks

In a similar manner, following the terminology defined in [9], the parameters used in the definition of the QoSConstraint expression called FlowSpec are each QoSCharacteristics that have their own quantifying dimensions and associated values. Since there is only one platform independent model, no metamodel framework is needed for the PIM.

4.4 Mapping strategy

The mapping layer appears between the PIM generic layer and PM specific layer. This layer is refined/modified for each specific network model it is targeting. The mapping will have both design time and runtime aspects. Comparing the model elements of the PIM and the PMs

specially IrDA and Bluetooth defined above, most of the model elements except the QoS related elements exist in both the PIM and the PMs. For such elements, the mapping is straightforward.

Therefore, the major mapping required is a QoS specification mapping. The parameters used in the PIM must be transformed to the parameters used in the PMs. Moreover, since the PIM model specifies a requirement specification and the PM specifies a provision specification (commonly a run time situation), verification between the required QoS and the provided QoS should be made even before the actual contract confirmations. Therefore, this approach also provides a mechanism for model analysis (at design time) and admission testing (at runtime).

The strategy divides the mapping into two basic parts:

I) **The Functional Service Elements**: It does a correspondence mapping between the PIM elements and PM elements, which match with each other. As described earlier, this is a relatively straightforward mapping. This includes all functional interfaces associated with message interchange.

II) **The QoS:** this does the mapping between the quality of service specifications. Since Throughput and Latency requirements are explicitly specified at the flow specification (PIM), the mapping is made separately for parameters related to these two QoS Categories. The parameters determining the two QoS characteristics should be identified from both the PIM and specially from the PMs.

It has two basic procedures:

Transform (adapt): which is responsible for actually converting the PIM parameters to PM parameters. This is used to initiate a reservation process depending on application request. The mapping layer mediates the application request and the platform provision.

Verify: which is responsible for verifying requests with provisions. Since the applications are basically soft real-time applications, this operation can be invoked at both design time (by using maximum and minimum possible provision values against a possible flow request) and runtime (by using the actual requests and provisions). Appropriate interaction interface functions must be defined to reflect the verification results. The application must be notified if the verification is successful or not.

A) Service Type mapping

The PIM specifies three types for this parameter namely, **Guaranteed**, **Controlled Load**, **Best Effort**. For this case, the support from the PM will be checked and accordingly, the

Throughput and Latency mappings will be made. Each network will have its own support. For example, there is no explicit representation for service types in IrDA, and Bluetooth supports Guaranteed, Best Effort, and None. Hence, appropriate interpretation of the PIM specification towards the PM specification is required.

B) Throughput related mapping

This mapping module transforms the parameters in the PIM that determine the throughput characteristics against the parameters that determine the throughput in the PM QoS specification. Here, we will use the concept of **Maximum Transmission Boundary (MTB)** that can be easily obtained from reservation-based specifications. MTB is the maximum amount of data that can be transmitted in a certain time interval T throughout the flow's lifetime.

The Maximum Transmission Boundary of a flow specification whose parameters are Token Rate (r), Bucket Size (B), MTU (M), and Peak Rate (p) is given by [12]:

$$MTB <= Min\,(B + rT,\ M + pT)$$

The parameters from the Platforms will be appropriately obtained and the corresponding MTB for the PM will be evaluated and it will be verified with the MTB of the PIM. Using maximum and minimum possible limits, it will be possible to make design time analysis and admission testing. The effective action will depend on the service level requested.

C) Latency related Mapping

The latency also can be mapped with appropriate boundary values, the maximum and minimum delay between packets will be explicitly specified in the PIM. The corresponding boundaries are computed from the platform level parameters and depending on the Service Level the necessary actions will be performed. In a similar manner to throughput related mapping, both design time and runtime aspects will be considered.

The problem with latency mapping is that in different networks the meaning of latency differs. For example in Bluetooth Latency is defined as *"The time from the delivery of a frame (bit) to the baseband layer until it is sent through the Radio"*. The time is determined by the polling interval currently enforced at the Baseband layer. According to the Bluetooth specification, the L2CAP layer can identify this value of latency as a maximum value. For IrDA delay is not explicitly specified but it can be estimated from the other parameters [52].

Delay has two parts: Fixed delay (Transmission Delay, etc) and Queuing delay. This is the interpretation given for Internet based flows. In our case, the fixed delay is composed of **Transmission Delay** over the physical medium (the radio or over the infrared link). The variable Queuing delay is the **Access Delay** whose factors are different in different networks. We will use this common interpretation for delay. Since the networks are short range, the transmission delay can be taken as some very small constant, which can possibly be ignored. For example in IrDA, the air transmission delay is 1/c, where c is the speed of light.

4.5 The Mapping Model

The negotiation between the application and the target network is made through the intermediate mapping layer, which is defined for that specific network forming a PSM.

Our idea is that the mapping layer will have both a design time and runtime version targeting a single network for mapping platform independent model and implementation with specific target platform model and implementation respectively. This will meet the objective of MDA in that communicable embedded applications can be designed and implemented without the concern of the peculiar characteristics of a used network.

This is possible as long as the appropriate mapping has been defined from the generic PIM level specification to that specific network service specification. Defining the mapping between the PIM and the PM is will not be basically a concern of the application developers.

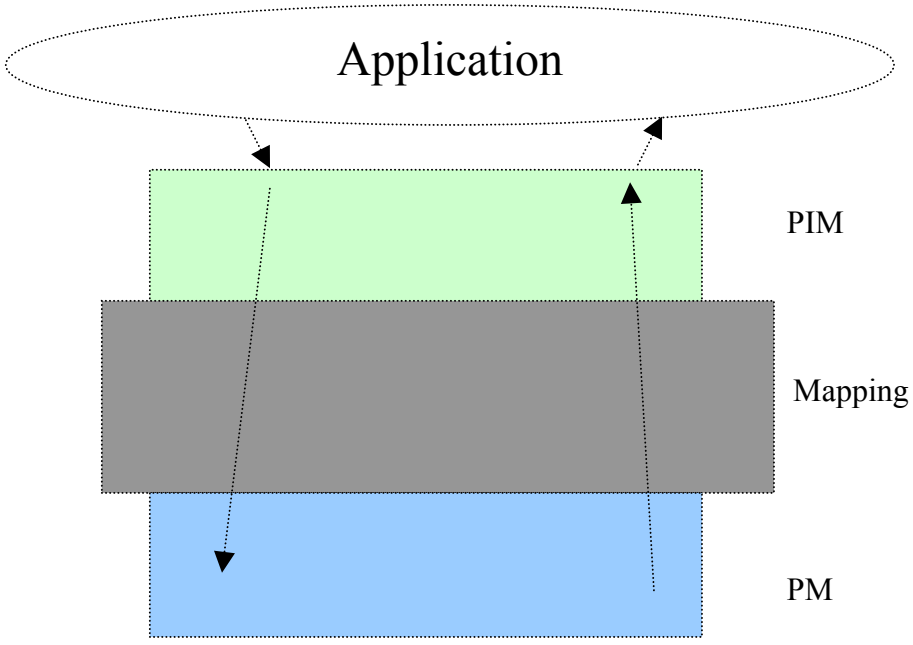

Figure 4-14: Conceptual Model of the Mapping

For modeling the mapping with UML, the two concepts, i.e., the Functional Service and the QoS are separated in to two groups (packages).

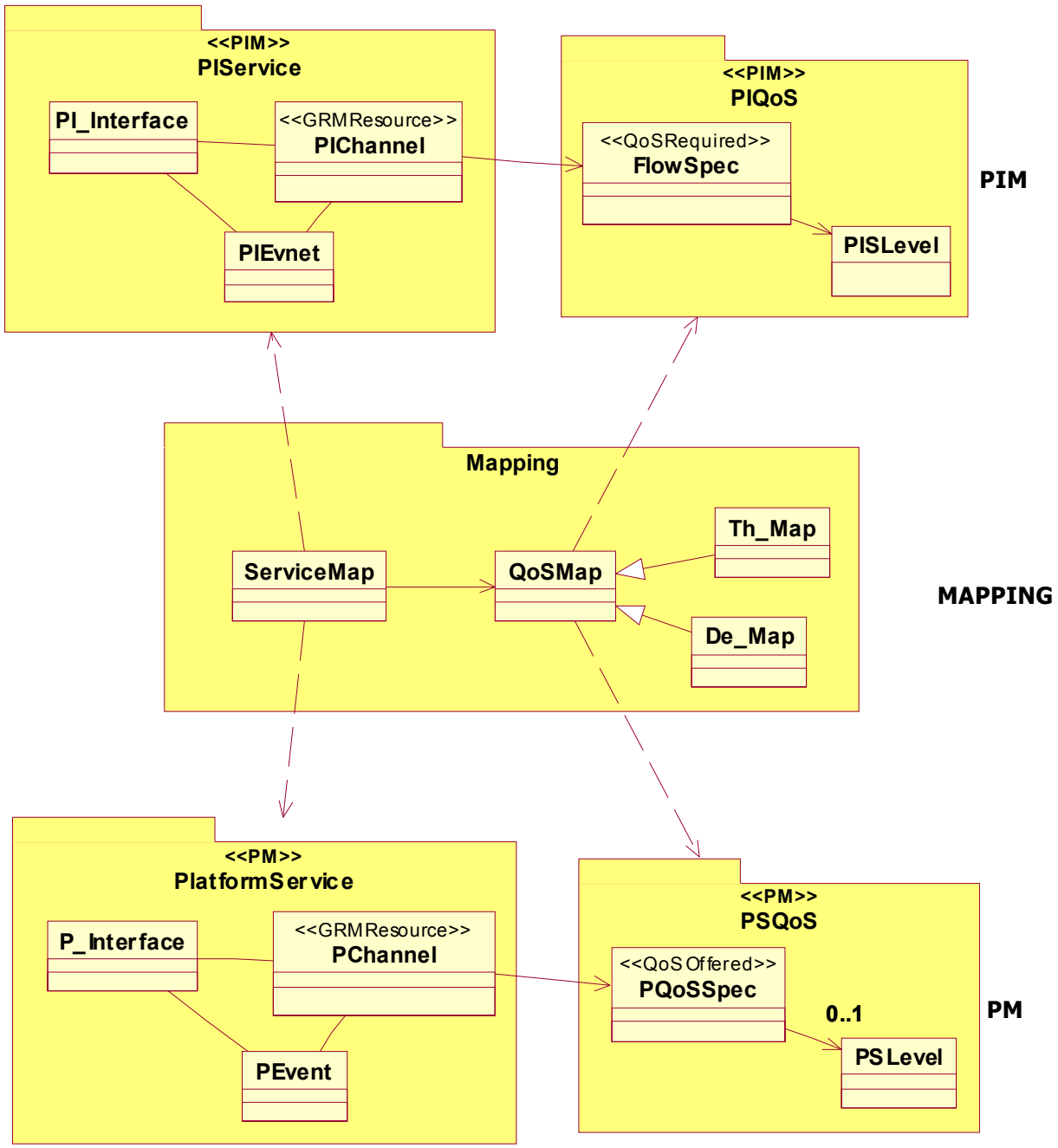

Figure 4-15: The actual proposed mapping model including the PIM and the PM in UML

The mapping layer will have the following elements:

ServiceMap: this is responsible for mapping the functional service interface from the PIM to the PM the operations defined in the PIM will be mapped with the operations with the PM. The abstract channel definition will be mapped to specific channel created by a specific network. Since there are several events generated, the mapping interface must be able to interpret the events appropriately in both directions.

QoSMap: The major mapping element is the QoS mapping which is responsible for Verifying the Required QoS (FlowSpec and Level) with the Offered QoS(QoS_Spec and Level) and transforming the parameters as described above. The main class QoSMap will have two subtypes related to Throughput related mappings (Th_Map) and delay related mappings (De_Map). This will create a flexible layer that links the generic flow based specification with different reservation based network platforms.

4.6 Conclusion and remarks

This chapter presented the main MDA based approach for embedded network services, which includes both functional and QoS characteristics. The QoS characteristics are based on performance factors namely throughput and delay (latency). Both a PIM and PM modeling techniques and a possible mapping strategy has been proposed. The major focus of the mapping is the Quality of service aspect.

Since there is no parameter for link error rate in both the Bluetooth and IrDA QoS specifications, it is not included in the model. The assumption is that a loss less or a minimal loss rate link is used. The link threshold time (link timeout time) used here is the maximum amount of time that forces an abnormal teardown of the link in case of permanent break of communication. But, this does not include packet loss rate and error correction tolerance.

It should be understood that the mapping layer does not represent an additional protocol layer. Instead it should be considered as a service layer that links an abstract network interface and a specific network interface. Its purpose may be related to that of a middleware. But the usage here is much different from the standard purpose of middleware. At the top it will have a generic interface and at the bottom, it will appear as a user of a specific network. This way, it separates the two concerns of development in this domain as Application and Network.

CHAPTER 5. Applicability of The Approach

5.1 Overview

In this chapter, the proposed approach will be verified with a mapping procedure to the logical models of Bluetooth and IrDA networks. Since the functional interfaces are relatively similar, the major focus of the mapping is the Quality of Service specifications.

As discussed in the proposed approach in the previous chapter, the QoS is modeled and implemented in the systems with two separate layers: the application level (PIM) layer and network level PM layer. The mapping layer exists between the two as a linking and mediating layer specially transforming the PIM level **QoS request** to the PM level **QoS provision.**

The application may incorporate the PIM in its communications subsystem, which will be implemented as a client for the mapping layer, which in turn is a client for the target network layer. The negotiation between the two peer application ends will be made through this level of three layers that work in coordination. After the QoS mapping, both the PIM and PM level QoS settings will be updated to reflect the negotiated parameters so that both the underlying network and the application will honor these QoS settings respectively.

The general procedure is described below:

> *The application initiates a communication and reservation request for a flow from one end to the mapping layer. The mapping layer computes (estimates) the major QoS requirement from this requirement and then computes the parameter values required from the underlying network. Then, it passes this request to the target network layer using appropriate target network parameters. The network will perform its typical negotiation process and notifies the mapping layer the actual reservation results. The mapping layer then verifies whether the provided reservation information agrees with the requested parameters for both throughput related and latency related requests. Then notifies the application the positive or negative results. The application is expected to interpret all interface events from the mapping layer and perform actions as appropriate to its objectives.*

The mapping layer computes the necessary QoS limits to transform and verify the appropriate parameters from the PIM (Flow) to the PM (Network specific).

5.2 Mapping to Bluetooth

In this section, how the mapping model can be defined for Bluetooth is presented. Fortunately, there is a similarity on the conceptual and operational QoS specification between Bluetooth and the PIM.

According to MDA, this is the best situation where some platform models resemble with the platform independent model so that the transformation process is very minimal. The mapping task is relatively straightforward. The only task that might be necessary is verifying whether the requested and the provided flow specifications match with each other. We will see how all the procedures are accomplished if Bluetooth is the target network. The mapping layer is customized to Bluetooth.

5.2.1 Mapping The Service Interface

The PIM interface and the PM interface have relatively similar architecture and elements and so the mapping can be simplified. The mapping layer simply links as a transparent layer between the two. Some improvement mechanism may be made to directly link the application with the platform level interface in such cases.

5.2.2 Mapping QoS

a. Service level

Below is a service level mapping from the PIM to the PM. How each service level is mapped as shown is further explained in the subsequent sections.

PIM	Guaranteed	Controlled Load	Best Effort
Bluetooth (PM)	Guaranteed	Best Effort	Best Effort

b. Throughput Related Mapping

At this point, the computation can be made with the same expression for both the PIM and PM. The **Maximum Transmission Boundary (MTB)** expression must hold for both of them where $MTB_{PIM} <= MTB_{PM(Bluetooth)}$. Before computing the requested and the provided MTB. The PIM parameters will be assigned to the PM (Bluetooth) parameters. Then the negotiation is initiated. After the response is received, the appropriate action can be performed depending

on the required **Service Level.** In this case, the following parameter interpretation must be made:

- The Guaranteed service level in the PIM enforces that at least the requested token rate to be assured. The platform must assure a token rate greater or equal to the requested token rate. More over it has to obey the MTB rule: $\mathbf{MTB_{PIM} <= MTB_{PM}}$**.** Otherwise the link cannot be made.
- If Controlled Load and BestEffort the two are taken as Best effort in Bletooth since there is no other level in Bluetooth. According to the flow specification [33], the two levels will be dealt with equivalently (i.e., as Best Effort) in many conditions such as the case in which the network is lightly loaded. In Buetooth, the Best Effort level requires as much reservation as possible that can be maintained at both ends. This characteristic is equivalent to the flow specification's Best Effort level.
- The 'None' setting in Bluetooth means No Traffic at all which doesn't have any equivalent at the PIM level. Since all flow specifications require at least the Best Effort level, this level will not have a value.

c. Latency Related Mapping

Since the delay is specified explicitly in both the PIM and the Bluetooth specification, a mapping can be easily made after estimating the transmission delay and link layer processing delay by the Bluetooth Link layer interfaces. The delay in the Bluetooth specification is the access delay on the sender's side, which does not include the transmission delay and the processing delay at the remote side.

By estimating these values (transmission delay and receiver's processing delay), the end-to-end delay can be computed. Then the verification and mapping can be made easily.

Strict verification is necessary if the service level is Guaranteed, for the other levels, it can be ignored to follow the working specification of a flow.

5.3 Mapping To IrDA

5.3.1 Mapping the Service Interface

Similar to the Bluetooth case, the PIM interface and the PM interface have relatively similar architecture and elements and so the mapping can be simplified. The mapping layer simply links as transparent layers between the two. Some optimization mechanism can be made to minimize the overhead imposed by the mapping layer.

5.3.2 Mapping QoS

a. Service Level

In the IrDA specification, there is no distinct expression for service levels. The most common scenario is that for the communication to begin between to application ends, both must agree upon the exchanged QoS parameters. The negotiation values are distinct enumerated values, which do not accept any other values. Furthermore, the two ends must honor the agreed upon values throughout the lifetime of the link. Hence, the best match for IrDA link layer service level for the PIM service level is the Guaranteed level. However, the application level specification still takes the three service levels. Therefore, the most appropriate accommodation in the link layer would be as follows.

If Guaranteed is requested, then strict parameters must be calculated and negotiated with the target IrDA link. The mapping layer then verifies the two values and decides on the success or failure.

If Best Effort or Controlled then the mapping layer will accept negotiated parameters whether they satisfy the requested values or not and notifies the resulting reservation values to the application. Then it will be up to the application to accept or reject the resulting values.

PIM QoS Specification is composed of:

Token rate (r)
Token Bucket Size (B)
Peak Rate (p)
Max Transmission Unit (MTU)
Delay

This should be mapped to the IrDA (PM) based QoS parameters which is composed of:

Baud Rate (Br)

Maximum Turn Around Time (MTt)

Data Size (DS)

Window Size (WS)

Link Threshold Time

Minimum Turn Around Time (mtt)

b. Throughput Related Mapping

Since the QoS negotiation parameters of the Flow Specification (PIM) and those of IrDA are different, the mapping requires an indirect but semantically similar mapping procedure.

The throughput based mapping can be made using the same concept of Maximum Transmission Boundary (MTB) using two parameters or IrLMP. According the IrDA specification [56,57], the Data rate (br) and Max. Turn around Time (MTt) have the highest priority in that the agreement on them influences the other parameters such as the Data Size and the Window Size which are major factors for throughput in IrDA.

Based on the IrDA QoS specification, the maximum amount of Data that can be transmitted in a single MTt is limited by the following expression.

$$\mathbf{MTB_{IrDA} <= Br * MTt} \qquad \mathbf{(1)}$$

This is the absolute maximum (i.e. not the actual maximum), the actual transmission in IrDA is calculated from the product of window size and the Data Size which agrees with (1).

The mapping mechanism can continue, after getting the flow characterization parameters. It can collect the available capacity of the device from the two key parameters: **MTt** and **Br**.

Based on this scenario, what the mapping layer can do is described below:

Since $\mathbf{MTB_{PIM} <= MTB_{PM}}$ should always be true, then the following inequality must hold (taking $MTB_{PIM} = B + MTt * r$ and $MTB_{IrDA} = MTt * Br$):

$$\Rightarrow \quad B + MTt * r <= MTt * Br$$

$$\Rightarrow \quad \mathbf{Br >= (B + MTt*r) / MTt} \qquad \mathbf{(2)}$$

If the second case is taken, i.e., if $MTB_{PIM} = M + p*MTt$, then

$$Br >= (M + p*MTt)/MTt$$

Among all IrDA parameters, the MTt has the highest priority, and it can be taken from the maximum setting that the PM can support, and the baud rate (Br) can be negotiated to satisfy the request.

If a Guaranteed request is received, then the minimum Br such that equation (2) holds true should be negotiated. If not, the mapping layer can notify the failure of the request.

For the other two types of negotiations, since relaxation is possible, the resultant negotiation will be taken and the application will be notified whether the request has got its target or less. The flow's rate can be calculated from the negotiated results with a simple reverse calculation. The application, according to its internal policy, will make its own decisions on the results.

For example, let's assume that new negotiated values appear to be MTt, Br at the platform level that could not satisfy the request. Then, for the PIM level, r or p can be calculated. Hence, we get:

r <= (B - MTt*Br)/MTt or

p <= (M - MTt*Br)/MTt , But in this case p must remain above r otherwise , the two parameters must be exchanged.

c. Latency Related Mapping

Delay is not explicitly described in the IrDA specification. However, from the work done in [52], the major delay factors for a frame at the link layer are: Minimum Turn Around Time (**mtt**), Queuing delay (**p**) at the source and the transmission delay (**Tt**), Acknowledgment delay (**tk**), and packet processing delay at the receivers side (**p**).

p = 4 * 10^3 / v , where v is the processor speed.

Tk = (2mtt +2p)/WS, acknowledgement is done for a window.

Tt = $\frac{Br}{DS}+\frac{1}{c}$, where **c** is the speed of light and 1 is the average distance between the devices (1 meter).

Therefore, the maximum delay that can be experienced by a frame, which is guaranteed by the negotiation parameters, on an IrDA link is given by:

$$\mathbf{D = Tt + Tk}$$

$$\mathbf{D} = \frac{Br}{DS}+\frac{1}{c}+\frac{2(mtt+ p)}{WS} \qquad \textbf{(3)}$$

Computing this value, the mapping layer can compare the requested delay for a packet with the maximum delay for a packet from the IrDA link. Then it reports the variation to the

application. It will be up to the application to decide continue the flow or cancel the connection. Taking a range of processor speeds, the value of packet delay can be estimated as bounded range of values.

5.4 Summary and Conclusion

This chapter has shown how the mapping can be made from the PIM to specific PSMs which is QoS enabled. Several other factors are not considered here, the OS efficiency, over all system capacity and other factors have not been included in this work. From the embedded system development experience, these factors are generally taken as bounded with in some values. Specially, the queuing (access) delay and data throughput are affected also by other factors. The mapping layer is also a process, which takes its own processing time, which will affect the overall throughput and generally the performance of the communication between the applications. Considering its task, i.e., mapping of the QoS parameters that is done before the flow begins it can be assumed that its effect on the actual flow will be insignificant.

CHAPTER 6. Conclusion and Future Works

6.1 Discussion

In essence, MDA is not a modeling language or standard by itself. But it is a methodology that integrates a set of modeling tools and standards for a new development approach. Although it has been introduced and tested in the enterprise systems, its concepts, goals and benefits make it very interesting to be adapted to other domains of system development that face similar or even worse levels of complexity such as the domain of embedded systems and more importantly the domain of pervasive systems since it makes extensive use of embedded devices and applications.

Platform independence from the enterprise point of view indicates that a model of system can be produced in such a way that it only describes its objective and structure with no implementation concepts. It does not include any information about what the model requires form an implementation platform. This case will be considered after a target platform has been identified and transformations are made.

The terms platform and abstraction are relative depending on how and where they are defined and used. A model at some abstraction (platform) level may be considered as platform independent for the abstraction levels below it. The common fact is that as we go down an abstraction hierarchy the following characteristics are observed:

- The models will include more details often referred as refinements.
- The number of choices (instances) of lower layer models that an upper layer abstraction model maps to will increase.
- Lower layer abstractions bring the models more close to actual implementations.

Therefore, the higher we go in the abstraction hierarchy, the more freedom we have for implementation choices and the lower we go, the easier we get the models for implementation.

The standard models incorporated in MDA specially the UML are extensible to address the peculiar modeling problems of embedded systems. The basic artifacts have been defined at the metamodel level using the UML profiles. What the profiles lack is that they don't show how to apply the concepts in an actual MDA based development process.

Embedded systems have been following a very different approach than enterprise systems. The case in embedded and real-time systems is different in that the systems are naturally close to platforms. There is no embedded system design without a platform artifact in it whether it is specific or general. Their design is highly influenced by a set of target platform models and capabilities. Hence, platform independence in embedded systems is that the system model targets a certain abstract platform that can support the required execution environment for the system. The "Abstract Platform" represents a set of services provided by a family of embedded platform architectures.

Resource constraints are major issues in this domain so that Quality of Service model that can resolve the question of efficient utilization of the platform capabilities for the required level of satisfaction required by the intended applications.

This thesis has introduced an adaptation of the MDA towards the development of embedded applications. It focused on the development of embeddable applications on communicating objects that make use of short-range networks for local area interaction. It used the recent proposals on Platform Based Development in combination with the MDA concepts. It also applied the UML profiles for defining the Resource and QoS artifacts in both the PM and the PIM.

We have proposed an approach based on the MDA vision of separating concerns of application systems specification and platform specification, as it should work for embeddable communicable applications of our time. This work argues that Platform Service Abstraction is a basis for adapting MDA in this domain so that applications are modeled and possibly implemented with a platform independent interface that will interact with a mapping layer, which takes the responsibility of linking the application with a specific model.

Generally, the Platform Specific Model is the combination of the Platform Model (PM) and the specific mapping layer targeting it.

The mapping layer is made to map (target) a single network at a time because what we are intending is a link between two specifications and not a full-fledged middleware layer. The devices are resource constrained in that a full-fledged middleware cannot be deployed on them. Each device must have both the mapping layer and the application implementations. Hence, the overhead imposed by the mapping layer must be reduced.

6.2 Thesis Contributions

Below is a list of basic contributions made by this work:

1. We believe that we have introduced an MDA based development strategy for embeddable communicable applications, which is useful beyond the concept of distributed embedded systems. Here, applications designed and deployed with this methodology for any purpose by different developers and possibly at different times can communicate. If they appear to come together (closer to each other) so that their mapping layers support the current underlying network between them, then they can communicate easily. This can be taken as one contribution towards the objective of future pervasive systems, i.e., deploying applications on all communicating objects.
2. The usage of reservation based specification for the communication subsystem of the application based on the token bucket based QoS requirement formulation is easy for applications. The mapping layer's model works with the predictable nature of both required and the provided QoS. It can neutrally verify and transform the requirement and the events generated both from the network and the application are mediated appropriately so that the separation of concerns is extended up to the runtime execution environment in addition to the design time analysis and testing possibility.
3. This approach can be taken as a sample on how to use the UML profiles in actual modeling instances such as this. We have applied the concepts of the in the QoS profile and the Resource profile to model both specific platforms and the PIM abstraction including the mapping model.

6.3 Future Works

In the future we have planned the following works

1. Because of time limitation and lack of tool support, the approach has not been implemented and tested with the actual networks. The evaluation presented in chapter 5, has shown the possibilities of mapping the QoS parameters between the application and a specific target network. But, it does not show the overhead imposed by the mapping layer. These issues are mostly experimental. It is our conviction that the approach will need to be extended and more refined to include all attributes, methods and operations for the modeling elements when tested with appropriate tools. This is one aspect that is left to be dealt with in the future.

2. Defining a similar approach for the other wing of networks is another aspect of future works. The other category of networks that employ priority based QoS provision and typically a connectionless service require a similar investigation for an MDA adaptation. This is also left as a future work.
3. There is a possibility that the mapping layer may be implemented for a family of networks, such as reservation-based networks as a whole, instead of each individual network as proposed in this model. Although this is related to device capacity, the mapping layer can incorporate libraries that can map the application request to a number of networks with little modification on the approach proposed in this work. This is one aspect that could be considered in the future.
4. Lastly, designing an integrated methodology that can incorporate both categories in an MDA standard is also another future aspect.

REFERENCES

1. M. H. Lee Model Based Reasoning: A principled approach, Software Concepts and tools – 2000) pp. 179-189
2. MDA Guide Version 1.0.1 (2003), OMG's official document:

   www.omg.org/docs/omg/03-06-01.pdf, Visited on Jan 10, 2005.
3. John D. Poole, Model Driven Architecture Vision, Standards and Emerging Technologies: Workshop on Metamodeling and Adaptive Object Models:

   www.cwmforum.org/Model-Driven%20Architecture.pdf, Visited on Feb 20, 2005
4. Artif Mashkor, Investigating the Model Driven Architecture (a Masters Thesis) – May 2004: http://www.cs.umu.se/education/examina/Rapporter/AtifMashkoor.pdf, visited on Mar 4, 2005.
5. Bechler et. al. Quality of Service in Mobile and Wireless Networks:The Need for Proactive and Adaptive Applications: http://csdl.computer.org/comp/proceedings/hicss/2000/0493/08/04938026.pdf, visited on May 30, 2005.
6. David E. Simon, An Embedded Software Primer, Addison Wesley, 1999.
7. Fernandes et. al., Model Driven Methodologies for pervasive information Development - Universidade do Minho, Protugal : www.di.uminho.pt/~mompes04/Papers/Fernandes.pdf, Visited on May 5, 2005.
8. Arnor et. al: "QoS Aware MDA" , SINTEF, Ericsson, Simula Research laboratory, http://www-verimag.imag.fr/EVENTS/2003/SIVOES-MDA/Papers/Solberg.pdf, visited on February 27, 2005
9. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms: OMG's Official Document, http://www.omg.org/cgi-bin/apps/doc?ptc/04-06-01.pdf, visited on Sep 01, 2005
10. UML Profile for Schedulability,Performance, and Time Specification: OMG's Official Document, http://www.omg.org/cgi-bin/apps/doc?formal/03-09-01.pdf visited on Sep 01, 2005.
11. Aniruddha S. Gokhale, Douglas C. Schmidt, Tao Lu, Balachandran Natarajan, Nanbor Wang: "CoSMIC: An MDA Generative Tool for Distributed Real- time and Embedded

Applications", Middleware Workshops 2003: www.cs.wustl.edu/~schmidt/PDF/mda_wkshp.pdf, visited on May 15, 2005

12. Sumithra B. & S.H Edwards- Applying Object-Oriented Techniques in Embedded Software Design: http://web-cat.cs.vt.edu/PEBB/CPES02-Bhakthavatsalam.pdf, visited on Dec 25, 2004
13. Jean-Philippe Babau, J. P. Tournier: Communication Protocol Evaluation for Embedded Systems HICIT'03 International Conference on Industrial Technology: http://www.ro.feri.uni-mb.si/ICIT03/tech_prog/prog_SS.shtml, visited on Dec 29, 2004
14. John A. Stankovic et al.- Strategic Directions in Real-Time and Embedded Systems- a presentation: sunset.usc.edu/~neno/cs589_2003/Week5b.ppt, Visited on May 30, 2005
15. Marco Sgroi *et. al.,* Embedded System Design Using UML and Platforms,., UC Berkeley, September, 2001:

    www-cad.eecs.berkeley.edu/Respep/Research/asves/paper2002/Rong_FDL02.pdf, visited on Jun 5, 2005
16. Alberto Sangiovanni Vincentelli. Defining Platform-based Design. EEDesign of EETimes, February 2002
17. Grant Martin, UML for Embedded systems specification and Design (IEEE document): http://ieeexplore.ieee.org/iel5/7834/21541/00998386.pdf
18. Model-driven architecture - a technical perspective. Technical Report ORMSC/2001-07-01, Object Management Group, 2001. Online: http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01, visited on Feb 15, 2005
19. Meta object facility specification - version 1.4. Technical Report formal/02-04-03, Object Management Group, 2002. Online: http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf, visited on Mar 12, 2005
20. Request for Proposal: MOF 2.0 Query/Views/Transformations RFP. Technical Report ad/2002-04-10, Object Management Group, 2002. Online: http://www.omg.org/docs/ad/02-04-10.pdf, Visited on Aug 15, 2005
21. UML profile for Enterprise Distributed Object Computing specification. Technical Report PTC/2002-02-05, Object Management Group, 2002. Online: http://www.omg.org/cgi-bin/apps/doc?ptc/02-02-05.pdf, visited on Jun 12, 2005

22. XML metadata language specification - version 1.2. Technical Report formal/02-01-01, Object Management Group, 2002. Online: http://www.omg.org/cgi-bin/apps/doc?formal/02-01-01.pdf, visited on Jun 12, 2005

23. Common warehouse metamodel specification - version 1.1. Technical Report formal/03-03-02, Object Management Group,2003. Online: http://www.omg.org/cgi-bin/apps/doc?formal/03-03-02.pdf, visited on Jun 12, 2005

24. Unified Modeling Language specification - version 1.5. Technical Report formal/03-03-01, Object Management Group, 2003. Online: http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf., visited on Aug 12, 2005

25. Klasse Objecten, What is the Model Driven Architecture, Klasse Objecten Inc, the Netherlands, www.klasse.nl/english/mda/mda-introduction.html, last visited on Oct 29, 2005.

26. Model Driven Architecture (MDA), Document number ormsc/2001-07-01, OMG Architecture Board ORMSC1, July 9, 2001, OMG official cite:

ftp://ftp.omg.org/pub/docs/ab/01-02-04.pdf: last visited on Oct 29, 2005

27. MDA Success Stories, OMG's Official Cite: http://www.omg.org/mda/ , visited on Jan 03, 2005.

28. Bluetooth SIG, Bluetooth Specification Document, Official website: www.bluetooth.com/pdf/Bluetooth_11_Specifications_Book.pdf, visited on May 5, 2005

29. Patric J. Megowan et. al., IrDA Infrared Communications: An Overview:

http://www.web-ee.com/primers/files/irda.pdf, visited on may 7, 2005

30. Vladimir Myslik, Introduction to IrDA, Hardware Server, 1998: http://www.hw.cz/english/docs/irda/irda.html, visited on Mar 15, 2005

31. Robert Bosch, CAN Specification, Version 2.0, 1991,: www.semiconductors.bosch.de/pdf/can2spec.pdf, visited on Feb 25, 2005

32. The I2C Specification, Version 2.1, January 2000: www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf, visited on Mar 02, 2005

33. Partridge C., "A Proposed Flow Specification", Internet Engineering Task Force, Request for Comments (RFC 1363), available at: http://www.ietf.org/rfc/rfc1363.txt, September 1992, visited on May 30, 2005

34. Javier Muñoz, Model Driven Development of Pervasive Systems, http://www.di.uminho.pt/~mompes04/Papers/Munoz.pdf, visited on Jan 15, 2005

35. P.Boulet et. al.,. MDA for SoC embedded systems design, intensive signal processing experiment. FDL03, 2003.

36. Jo˜ao Paulo Almeida et. al., Handling QoS in MDA: A Discussion on Availability and Dynamic Reconfiguration, CTIT Technical Report TR–CTIT–03–27, University of Twente.

37. Sekaran K.C., Development of a Link layer protocol using UML, Proceedings of IEEE international Conference on Computer Networks and Mobile Computing, October 2001

38. K. Thramboulidis, A. and A. Mikiroyannidis, Using UML for the Design of Communication protocols: The TCP Case study, 11th International Conference on Software Telecommunicationand Computer Networks, October 7-10, 2003

39. Frolund, S. ; Koistinen, J.: QML: A Language for Quality of Service Specification / HP Labs. 1998 (TR-98-10). Technical report

40. Aagedal, J. Ø.: Quality of Service Support in Development of Distributed Systems. Oslo, Norway, University of Oslo, PhD. Thesis 2001

41. W. Torben, Model-Driven Development of QoS Enabled Distributed Applications, University of Electronics and Information Technology, Berlin, PhD thesis 2004: http://edocs.tu-berlin.de/diss/2004/weis_torben.pdf, visited on May 31, 2005

42. Jan Øyvind Aagedal and Earl F. Ecklund, Jr., Modelling QoS: Towards a UML Profile: Presented at Fifth International Conference on the Unified Modeling Language -the Language and its applications (October 2002): http://www.heim.ifi.uio.no/~janoa/papers/UMLProfileQoS.pdf, visited on May 25, 2005

43. Alberto Sangiovanni Vincentelli et.al., Defining Platform-based Design. EEDesign of EETimes, February 2002

44. Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Printice-Hall, 2003.

45. S. Shenker, J. Wroclawski, Network Element Service specification Template, RFC 2216, 1997: www.rfc-archive.org/getrfc.php?rfc=2216, visited on Jun 20, 2005

46. Selic, B., “A Generic Framework for Modeling Resources with UML,” IEEE Computer (vol. 33 no.6), June 2000 (pp. 64-69).

47. John A. Stankovic et al.- Strategic Directions in Real-Time and Embedded Systems- a presentation: sunset.usc.edu/~neno/cs589_2003/Week5b.ppt, Visited on May 30, 2005

48. ARTIST- Adaptive real-time systems for quality of service management- Roadmaps for Research Draft IST-2001-34820, 2003, official site: http://www.artist-embedded.org/, visited on Jan 20, 2005.

49. S. Shenker et. al., Specification of Guaranteed Quality of Service (RFC 2212),Network Working Group 1997

50. R. Canonico et. al., On the introduction of QOS awareness in legacy distributed applications, Proceedings of the 14th international conference on Software engineering and knowledge engineering (659-664), July 15-19, 2002, Ischia, Italy. ACM, 2002

51. Marko Hännikäinen(PHD), Design of Quality of Service Support for Wireless Local Area Networks Tampere University of Technology, 2002

52. IrDV. Visttas and A.C Boucouvalas, IrDA IrLAP protocol perfomance and Optimum Link layer parameters for maximum throughput, Proceedings of IEEE GLOBECOM 2002,Tiapei.

53. Guruduth Banavar et. al., Challenges: An Application Model for Pervasive Computing, Proceedings of the Sixth Annual ACM/IEEE International Conference on Mollie Computing and Networking (Mobicom 2000)

54. H Xiao et. al., Flexible quality of service model for mobile Ad-hoc networks, IEEE VEHcle TECHNOLogy CONFerence, 2000, Tokyo Japan

55. Lee, S.B., Gahng-Seop, A., Zhang, X., and A.T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks", Journal of Parallel and Distributed Computing (Academic Press) , Special issue on Wireless and Mobile Computing and Communications, Vol. 60, No. 4, pp. 374-406, April 2000

56. Andy Seaborne et.al., Infrared Data Association Link Management Protocol Specification, Version 1.2, Jan 23, 1996: http://www.www.irxon.com/download/IrLMP11.PDF, visited on Jun 1, 2005

57. Timothy Williams et. al., Infrared Data Association Link Access Protocol Specification, Version 1.1, Jun 6, 1996: http://www.waterwood.com.cn/technology/irda_documents/IrLAP11.pdf, visited on Jun 1, 2005

58. Bernd Bruegge and Allen H. Dutoit, Object-oriented Software Engineering: Conquering Complex and Changing Systems, Prentice Hall, 2000.

ANNEX A

Bluetooth configuration primitive and its parameters

The Configuration Primitives

The service primitives of the Bluetooth channel configuration process are used for the purpose of reserving resources on the end devices before an intended send / receive activity are given below:

L2CA_ConfigReq (*INPUT*: CID, InMTU, OutFlow, OutFlushTO, LinkTO,
OUTPUT: Result, InMTU, OutFlow, OutFlushTO)

The use of this primitive requests the initial configuration (or reconfiguration) a channel to a new set of channel parameters. Input parameters are the local CID endpoint, new incoming receivable MTU (InMTU), new outgoing flow specification, and flush and link timeouts. Output parameters composing the L2CA_ConfigCfm(Neg) message are the Result, accepted incoming MTU(InMTU), the remote side's flow requests, and flush and link timeouts.

Input parameters:

CID: Local channel endpoint ID

InMTU: the MTU the initiator can receive

OutFlow: the QoS flow specification supported by the initiator

OutFlushTO: Sent packet Time out

LinkTO: Link Timeout

Output parameters:

Result: Success or Failure

InMTU: the MTU the initiator can receive

OutFlow: the QoS flow specification supported by the initiator

OutFlushTO: Sent packet Time out

LinkTO: Link Time out

These parameters are corresponding response parameters that will be received with a positive or negative response from the other End.

L2CA_ConfigRsp(*INPUT*: CID, outMTU, InFlow,

***OUTPUT*: Result)**

The use of this primitive issues a response to a configuration request event indication. Input parameters include the local CID of the endpoint being configured, outgoing transmit MTU (which may be equal or less to the OutMTU parameter in the L2CA_ConfigInd event) and the accepted Flowspec for incoming traffic. The output parameter is the Result value.

Input parameters

CID: Local channel endpoint ID

outMTU: the inMTU the initiator must expect

OutFlow: the QoS flow specification supported by the responder

Output parameters:

Result: the result indicates whether the configuration has been accepted or not.

These parameters are corresponding response parameters that will be received with a positive or negative response from the other End.

L2CAP QUALITY OF SERVICE (QOS) OPTION

This option specifies a flow specification (flowSpec) similar to RFC 1363 with some modifications. The out **InFlow** and **InFlow** parameters in the configuration primitive are defined using this QoS option. If no QoS configuration parameter is negotiated the link should assume the default parameters discussed below. When included in a Configuration Request, this option describes the outgoing traffic flow from the device sending the request to the device receiving it. When included in a positive Configuration Response, this option describes the incoming traffic flow agreement as seen from the device sending the response. When included in a negative Configuration Response, this option describes the preferred incoming traffic flow from the perspective of the device sending the response.

Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort must be assumed. If any QoS guarantees are required then a QoS configuration request must be sent. The remote device places information that depends on the value of the result field, its Configuration Response. If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is “Failure – unacceptable parameters”, the response may

include a list of outgoing flowspec parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response.

Service type (Guranteed, BestEffort, None)

Token Rate (Bytes/Second)

Token Bucket Size (bytes)

Peak Bandwidth (bytes/second)

Latency (microseconds)

Delay Variation (microseconds)

Service Type:

This field indicates the level of service required. If 'None' 'No traffic' is selected, the remainder of the fields may be ignored because there is no data being sent across the channel in the outgoing direction.

If 'Best effort', the default value, is selected, the remaining fields should be treated as hints by the remote device. The remote device may choose to ignore the fields, try to satisfy the hint but provide no response (QoS option omitted in the Response message), or respond with the settings it will try to meet.

Token Rate

The value of this field represents the rate at which traffic credits are granted in bytes per second. An application may send data at this rate continuously. Burst data may be sent up to the token bucket size (see below). Until that data burst has been drained, an application must limit itself to the token rate. The value 0x00000000 indicates no token rate is specified. This is the default value and implies indifference to token rate. The value 0xFFFFFFFF represents a wild card matching the maximum token rate available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

Token Bucket Size

The value of this field represents the size of the token bucket in bytes. If the bucket is full, then applications must either wait or discard data. The value of 0x00000000 represents no token bucket

is needed; this is the default value. The value 0xFFFFFFFF represents a wild card matching the maximum token bucket available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum buffer space available at the time of the request.

Peak Bandwidth:

The value of this field, expressed in bytes per second, limits how fast packets may be sent back-to-back from applications. Some intermediate systems can take advantage of this information, resulting in more efficient resource allocation. The value of 0x00000000 states that the maximum bandwidth is unknown, which is the default value.

Latency

The value of this field represents the maximum acceptable delay between transmissions of a bit by the sender and its initial transmission over the air, expressed in microseconds. The precise interpretation of this number depends on the level of guarantee specified in the Class of Service. The value 0xFFFFFFFF represents a do not care and is the default value.

Delay Variation

The value of this field is the difference, in microseconds, between the maximum and minimum possible delay that a packet will experience. This value is used by applications to determine the amount of buffer space needed at the receiving side in order to restore the original data transmission pattern. The value 0xFFFFFFFF represents a do not care and is the default value.

ANNEX B

The IrDA QoS Negotiation parameters

There are six major QoS negotiation parameters specified in the QoS specification used at connection setup. The negotiation parameter specification for parameters is a three-filed format containing the Parameter Identifier (PI), the Parameter Length (PL – in bytes), and the parameter value (PV) respectively. All parameters have distinct negotiable values that should be specified in the PV field as a bit pattern representing each supported value for the parameter by the device. Non-supported values will be presented as 0 and supported values are presented as 1. Generally, the parameters are devied in to two types Type0 and Type1. Type0 parameters must be negotiated to the same value by both devices and Type1 parameters may be negotiated independently for each device.

PI	PL	PV (bit pattern)

Baud Rate (Type0)

This parameter defines the data rate that should be used to send information by both devices. This parameter has 9 alternatives: 9600, 19200, 38400, 57600, 115200, 576000, 1152000, 4000000, and 16000000. It has to be negotiated to the same value by both ends. It has the highest priority next to the maximum turn around time. It is specified in **Bits per Second.**

Maximum Turn Around Time (Type1)

This parameter specifies for how long a station can hold the link sending data before handing over the link to the other device. Together with the Baud Rate, it determines the maximum amount bytes a station can send with in a Turn Around period. This parameter has four different alternatives. Specified in **milliseconds**. The Options are 500, 250, 100, 50 represented by bit0 to bit4 respectively in the PV field. It has higher priority than Data Size.

Data Size (Type1)

This parameter determines the maximum size of data **bytes** that can be received in a frame for the duration of the connection. This is the amount of the relevant data field in the

information (I) frames. It does not include protocol bytes. It has six alternatives: 64, 128, 256, 512, 1024, and 2040 represented by bits 0 to 6 respectively. Its value is adjusted to accommodate the Baud Rate and Maximum Turn Around Time.

Disconnect/ Threshold Time (Type0)

This parameter is used to control the time a station will wait without receiving valid frames before it disconnects the link. It had eight alternative values expressed in **Seconds**: 3, 8, 12, 16,20, 25, 30, 40.

Window Size (Type1)

This represents the amount of Buffer size that can prepare to receive information frames before acknowledgement. It is expressed by the number of Frames and has 7 alternative values: 1 to 7. The value of this parameter is determined by the values of Max. T. A. time, Baud Rate and the Data Size. It is not an essential parameter at IrLMP layer since it is calculated by the IrLAP layer mechanism. Although this value cannot be changed by a request from an IrLMP layer, application can request its initial value in the negotiation process.

Minimum Turn Around Time (Type1)

This parameter deals with the time needed for a receiver node to recover following saturation by transmission from the same device after receiving the last byte. It is also called **Turn Around Latency**. It has 8 alternatives expressed in **milliseconds**: 10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0. This is also used by the IrLAP layer and is not a relevant parameter for the IrLMP layer.

DECLARATION

The thesis is my original work and has not been presented for a degree in any other university, and that all sources of material used for the thesis have been duly acknowledged.

Eyob Alemu

Dr. Dawit Bekele (Advisor)