

Defining UML Profiles and Model Mappings in the context of MDA

Juan Pablo Zamora Zapata¹, Francis Bordeleau¹

¹School of Computer Science, Carleton University, Ottawa, ON, Canada
{jpzzapat, francis}@scs.carleton.ca

1 Introduction

The Object Management Group (OMG) is embracing Model Driven Architecture (MDA) as its newest initiative. MDA promotes the use of models and model transformations [1]. MDA's goal is to enable system model transformations from a Platform Independent Model (PIM) level to Platform Specific Model (PSM) level. A typical process may combine the use of several PIMs and PSMs to go from requirements to specific implementations. In such a process, the systematic model transformations are conducted using traceable model mappings. Model mappings are one of the cornerstones of MDA as they provide reusable entities that can be used by the end-users to go between different levels of model abstraction. Well-defined model mappings can be implemented to automate model transformations. MDA proposes the use of the Unified Modeling Language (UML) for language profile definition, system model creation and the definition of transformation models.

The Software Radio Domain Special Interest Group (SWRadio DSIG) [2] of the Object Management Group (OMG) is currently defining the specification of a new generation of radio, called Software Defined Radio (SDR). Following the OMG MDA approach, the SWRadio DSIG defines its specification in the form of a UML Platform Independent Model (PIM). The SWRadio PIM in its current state constitutes a specification for product development. This specification is essentially composed of a set of class diagrams, which describe the relationships between the classes that compose an SDR and the required interface of those classes. It also contains some use cases, sequence diagrams, and simple state machines that are informally used to complement the class diagrams.

The overall objective of our project is to define a systematic and traceable approach to go from the UML specification model defined by the OMG SWRadio DSIG to a UML Rational Rose RealTime (RRRT) design model tool. The choice of RRRT for design is based on the fact that RRRT supports some of the most fundamental concepts of MDA, including model executability and code generation, and that RRRT has been used for real-time embedded systems development in several application domains including aerospace, telecommunication, and defense. The RRRT notation is defined using a set of UML stereotypes defined in [3] and supported by the RRRT tool. The main RRRT stereotypes include capsules, protocols, ports and connectors.

In this project, we define a mapping that allows producing a RRRT design model from the SWRadio UML specification model in a systematic manner. This mapping is formally defined between two UML profiles: the UML Spec profile, which contains the set of UML concepts used in the SWRadio specification, and the UML RRRT Design profile, that contains the set of RRRT concepts. At a detailed level, the profile mapping is defined in terms of a set of element mappings, which define mapping between the high-level constructs used at the specification level of the Software Radio PIM and the notation used by the RRRT tool.

In spite of the fact that this work is conducted in the context of SWRadio, in this paper, we use an ATM system as an example to illustrate our approach. The reason is that we want to focus on the profiles and mappings. The use of the SWRadio model would require too much background explanation, and would decrease the readability of the paper for people not familiar with SWRadio.

The rest of the paper is structured as follows. Section 2 presents a brief description of the UML extension mechanisms and describes the two UML profiles we use in our project. Section 3 describes the process for mapping definition. We conclude in section 4 with a short summary of the paper.

2 Profiles

UML provides built-in *profile* facilities that allow for the tailoring of UML for different contexts. For example, UML profiles can be defined for different platforms (such as CORBA, J2EE or .NET) or domains (such as real-time or business process modeling). Through the UML profile facilities one can reuse, extend or redefine UML concepts (metaclasses), and essentially create new UML dialects [4][5] that fit the needs of a specific application domain.

The definition of a UML profile involves three main steps

1. Inclusion of UML metaclasses (or UML metamodel packages) that can be used within the profile.
2. Creation of new model elements defined as extensions of UML metaclasses.
3. Definition of constraints on the use of model elements and their relationships

The basic mechanisms to create UML Profiles are *stereotypes* and *tagged values*. Stereotypes are used to create new model elements, as extensions of UML metaclasses (step 2). In a class diagram, the name of a stereotype is shown within a pair of guillemets (e.g. <<stereotype>>) above the class name. The definition of stereotypes may also involve the definition of *tagged values* that can be used to define properties of the stereotype, and the definition of *constraints* that can be used to formalize certain aspects of the stereotype (step 3).

In the next two subsections, we introduce the *UML Spec profile* and the *UML RRRT Design profile* that we use in the SWRadio MDA project.

2.1 UML Spec Profile

The UML specification model, defined by the SWRadio DSIG, uses a subset of the UML notation. The UML Spec profile explicitly captures this subset.

Figure 1 a) illustrates the set of UML model elements contained in the UML Spec profile. This set includes classifiers of types Class and Interface, and relationships of types Realization, Composition, Dependency, Aggregation, and Association.

The *ATMSystem* of Figure 1b) is defined using the UML Spec profile. It defines the relationships between the different classes that compose an ATM system. There is a composition relationship between *ATMSystem* and the classes *CashDispenser*, *Display*, *KeyPad* and *Controller*, in which all of them share the same lifetime as the *ATMSystem*. Within the *ATMSystem*, two classes are related to a common *KeyPadInterface*. The *KeyPad* realizes the *KeyPadInterface* while the *Controller* uses it. With this definition, we can infer that there is communication between the *Controller* and the *KeyPad* even if it is not explicitly indicated. The *Controller* class also communicates with the *CashDispenser* and *Display*. The relationship is expressed through associations between the *Controller* with the *CashDispenser* and *Display* classes.

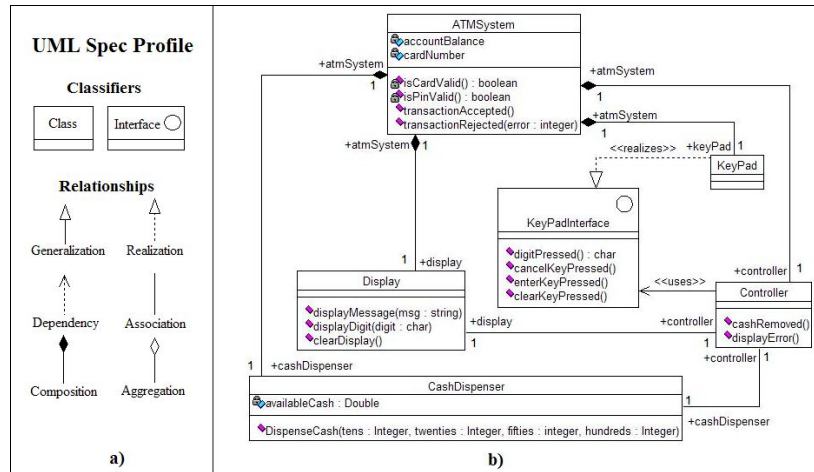


Figure 1. a) UML Spec profile Model Elements b) ATM UML Spec example

The model presented in Figure 1 b) illustrates two different approaches to define interaction between two structural elements. The first approach is illustrated using association relationships (*Controller* and *CashDispenser*). The second approach is illustrated by an interface definition (*KeyPadInterface*) and the realization and use of the interface by the *KeyPad* and *Controller* respectively. In the next section, we present those different approaches integrated into a concrete communication mechanism using the UML RRR Design profile.

2.2 UML RRR Design Profile

RRRT is a UML modeling tool for real-time embedded systems. It is used for the development of industrial embedded systems in different application domains including aerospace, telecommunication, and defense. RRRT allows for automatic code generation and requires precise semantics in the set of model elements it uses. The UML RRR Design profile contains the set of concepts used in the RRRT tool.

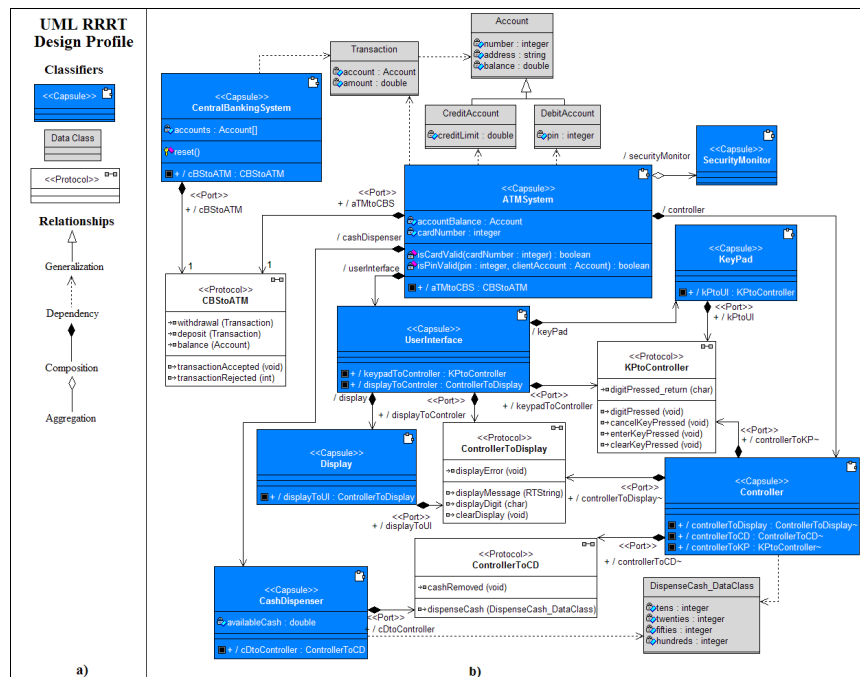


Figure 2. a) UML RRR Design profile model elements b) ATM UML RRR Design example

Figure 2 a) presents the UML model elements of the UML RRRT Design profile. Classifiers include Data Class, Capsule and Protocol. Relationships include Composition, Dependency and Aggregation. Association relationships are not used in the UML RRRT Design Profile because their semantics can derive in different interpretations thus lacking the precision required for UML RRRT Design model elements.

Figure 2 b) presents an ATM UML RRRT Design class diagram as the target model for our transformation example. The model elements used in the diagram are model elements defined in the UML RRRT Design profile.

3 Model Mapping

We create a mapping to make elements on our source models drive the construction of our target models. We associate elements on the source profile with elements on the target profile. We also define the sequential steps to execute the mapping and, if needed, we add constraints to it to ensure that the mapping will be applied only under some specific conditions. We extend the transformation metamodel defined in the OMG Common Warehouse Metamodel Transformation metamodel [6].

Because of a lack of space, in this paper we give an overview of the mapping we defined in the two profiles. Details of the mapping are omitted. The complete mapping between the UML RRRT Design profile and the UML Spec profile is defined in [7].

Figure 3 shows the properties and associations of a Mapping. The mapping specifies the model elements that will act as the source and target model elements in the transformation. The properties *function* and *functionDescription* include information on how to execute the mapping. The *constraints* property is used to define specific conditions (if needed) required for the execution of the mapping.

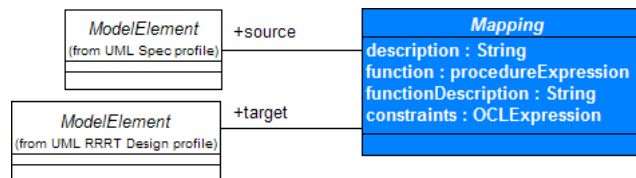


Figure 3. Mapping Definition.

In our example, we use a specification level model as our source model and a design level model as our target model. The mapping allows going from a specification model to a RRRT Design model in a systematic and traceable manner.

The definition of a mapping is performed in three main steps:

1. Analysis of the concepts defined in the profiles.
2. Definition of conceptual relationships between the elements of the profiles. Informal identification on how each model element in the source profile can be associated with a model element from the target profile
3. Formalize the relationships in a well-defined Profile Mapping package. Embed the element's association, traceable naming and constrain definition (if needed) in the definition of a set of mapping stereotypes.

Table 1 illustrates the result of steps 1 and 2 applied to the UML Spec and UML RRRT profiles. Direct associations are done with the Class classifier and with several relationships. New types of classifiers were created. Capsule and Protocol are specializations of the classifier Class and defined as stereotypes using UML extension mechanisms. Protocol Class is a specialization of a Collaboration and also defined as a stereotype.

Model Element	Graphical Representation	Supported by	
		UML Spec	UML RRRT Design
Classifiers:			
Class		✓ (Active and Data Classes)	✓ (Data Classes)
<<Capsule>>			✓
Interface		✓	
<<Protocol Class>>			✓
<<Port>>			✓
Relationships:			
Generalization		✓	✓
Realization		✓	
Composition		✓	✓
Dependency		✓	✓
Aggregation		✓	✓
Association		✓	

Table 1. UML Spec and UML RRRT Design model element association.

Figure 4 presents two examples of model element mapping between the two profiles (step 3). In both cases we present model elements of the UML Spec profile that maps to model elements from the UML RRRT Design profile. Figure 4 a) defines a mapping between an interface of the UML Spec and a Protocol Class in the UML RRRT Design profile. Figure 4 b) defines a mapping between a class of the UML Spec profile and two possible UML RRRT Design model elements: a Data Class or a Capsule. In this case, additional information needs to be provided in the form of a constraint within the stereotype. The stereotype definition needs to clearly specify when a UML Spec Class will be mapped into a UML RRRT Design Data Class and when it will be mapped into a UML RRRT Design Capsule.

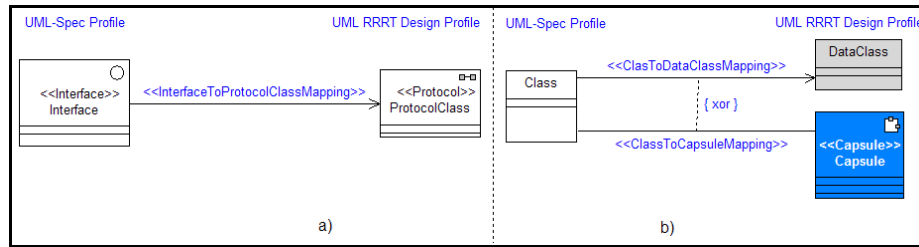


Figure 4. Model element mapping. a) Interface mapping b) Class mapping

The definition of the *ClassToCapsuleMapping* ClassifierMapping is shown in Figure 5. We highlight two properties of the *ClassToCapsuleMapping* definition:

- 1) The *function* property associates the name of the source class with the name of the Capsule to ensure traceability
- 2) The *constraints* property ensures that only active classes from the original UML Spec model are transformed into Capsules in the UML RRRT Design model.

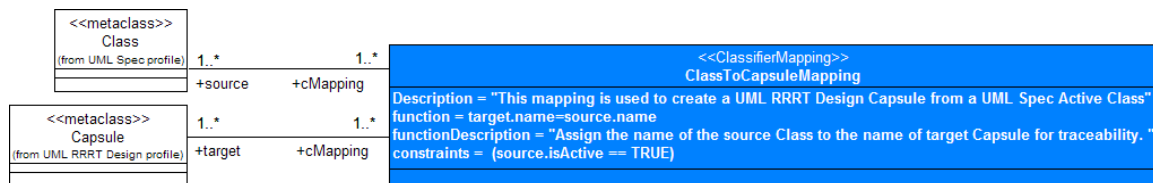


Figure 5. ClassToCapsuleMapping specification.

A more detailed description of how a UML Spec Class from Figure 4 b) is mapped onto a Data Class or a Capsule is illustrated in Figure 6. The mapping defines how the classifier is mapped and how the features within the classifier are mapped.

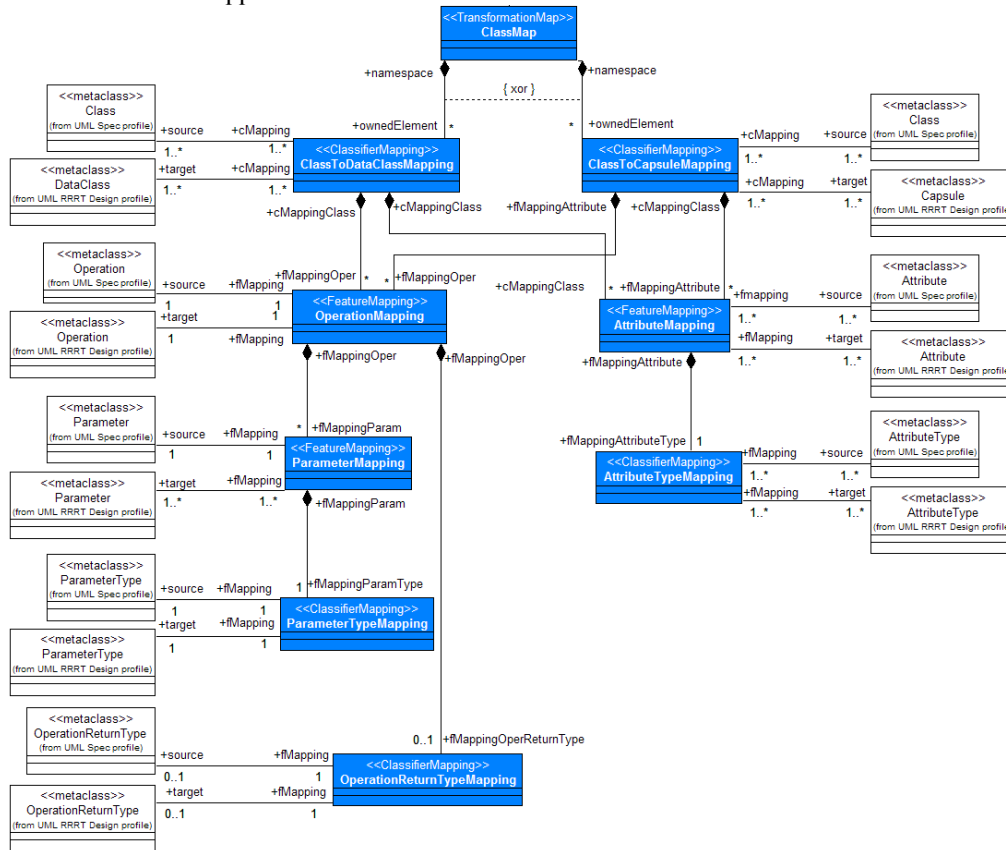


Figure 6. UML Spec to UML RRRT Design Class Map Overview

4 Conclusion

This short paper describes the two UML profiles that we use for the definition of a specification-to-design mapping in the context of the OMG SWRadio standardization effort. We also introduce the concept of UML model mapping and outline the mapping we defined between the UML RRRT Design profile and the UML Spec profile. This mapping allows making the transition between a UML spec and a UML RRRT Design in a systematic and traceable manner. The approach presented in this paper is model driven and follows the guidelines defined by the OMG’s Model Driven Architecture.

5 References

1. OMG Architecture Board. *Model Driven Architecture*. OMG whitepaper: <http://www.omg.org/docs/ormsc/01-07-01.pdf>
2. OMG SWRadio Domain Special Interest Group. <http://swradio.omg.org>
3. B. Selic, J. Rumbaugh, *Using UML to Model Complex Real- Time Systems*, Rational whitepaper: <http://www.rational.com/media/whitepapers/umlrt.pdf>
4. David S. Frankel. *Applying MDA to Enterprise Computing*. OMG Press 2003.
5. Grady Booch et al. *The Unified Modeling Language User Guide*. Addison Wesley. 1998
6. OMG. *Common Warehouse Metamodel (CWM) Specification*. www.omg.org.
7. Juan Pablo Zamora Zapata. *From Specification level to Design level: A Model Driven Architecture Transformation Model*. Ph. D. Thesis, Carleton University, Ottawa, Ontario, Canada. In progress.