UML Model Mapping in the Context of Software Radio

Juan Pablo Zamora Zapata[1], Francis Bordeleau[1], Jeff Smith[2]

[1] School of Computer Science. Carleton University. Ottawa, ON. Canada
{jpzzapat, francis}@scs.carleton.ca
[2] Mercury Computer Systems, Chelmsford, MA. U.S.A
jesmith@mc.com

Abstract. The Software Radio Domain Special Interest Group (SWRadio DSIG) of the Object Management Group (OMG) is currently defining the specification of a new generation of radio, called Software Radio. Following the OMG MDA approach, the SWRadio DSIG defines its specification in the form of UML Platform Independent Model (PIM). This paper defines a systematic and traceable mapping to go from the UML specification model defined by the OMG SWRadio DSIG to a UML Rational Rose RealTime (RRRT) design model. The 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. The transformation approach is based on the OMG Common Warehouse Metamodel (CWM) Transformation Metamodel.

1. Introduction

The Object Management Group (OMG) is embracing Model Driven Development (MDD) in its newest initiative: Model Driven Architecture (MDA). 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. Typical process may combine the use of several PIMs and PSMs to go from requirements to specific implementations. In such 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 models defined at different levels of 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) of the Object Management Group (OMG) is currently defining the specification of a new generation of radio, called Software Radio. Following the OMG MDA approach, the SWRadio DSIG defines its specification in the form of 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 the SWRadio 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. 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 domain including aerospace, telecommunication, and defense. The RRRT notation is defined using a set of UML stereotypes defined in [2] and supported by the RRRT tool. The main RRRT stereotypes include capsules, protocols, ports and connectors. These concepts are now included in UML 2.0 [3].

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 constructs used at the specification level of the Software Radio PIM and the notation used by the RRRT tool. In this paper, we focus on the mapping of the structural elements of the two profiles. The behavioral aspect is not addressed.

For sake of understandability, in this paper, we use an ATM system as an example to illustrate the mapping between the UML Spec and UML RRRT Design profiles. The use of the SWRadio specification 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. Section 4 describes the process for profile-to-profile model mapping. Section 5 illustrates an example of profile-to-profile model mapping using ATM models. We conclude in section 6 with a short summary of the paper.

2. Profiles

UML provides built-in profiling 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 profiling facilities one can reuse, extend or redefine UML concepts (metaclasses), and essentially create new UML dialects [5] that fits the needs of a specific application domain.
The definition of a UML profile involves four main steps

1) Inclusion of UML metaclasses (or UML metamodel packages) that can be used within the profile.
2) Creation of new model elements that result of the semantic modification of existing UML metaclasses to fit the requirements of the new profile.
3) Creation of new model elements defined as extensions of UML metaclasses.
4) 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 3), or to modify the definition of existing ones (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 4).

In this section, we introduce the *UML Spec profile* and the *UML RRRT Design profile* that we use in the SWRadio MDA project. The relationship between the ATM system model defined using the UML Spec profile and the ATM system model defined using the UML RRRT Design profile is discussed in section 5.

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 g*eneralization*, r*ealization*, d*ependency*, a*ssociation, composition and aggregation.*

The *ATMSystem* of Figure 1 b) is defined using the UML Spec profile. It defines the relationships between the different classes that compose an ATM system. Two types of Classifiers are shown: Classes (*Transaction, Account, CreditAccount, DebitAccount, CentralBankingSystem, ATMSystem, SecurityMonitor, Display, CardReader, KeyPad* and *Controller*) and an Interface (*KeyPadInterface*). Relationships in the diagram include generalization, realization, dependency, association, composition and aggregation. The diagram shows interaction at two different level of abstraction: at a high level, it shows interaction between subsystems (*CentralBankingSystem* and *ATMSystem*); at a lower level, it shows interaction between components that constitute the *ATMSystem* (*SecurityMonitor, Display, CardReader, KeyPad* and *Controller*).

Additional description of Figure 1 b) is provided in section 5.

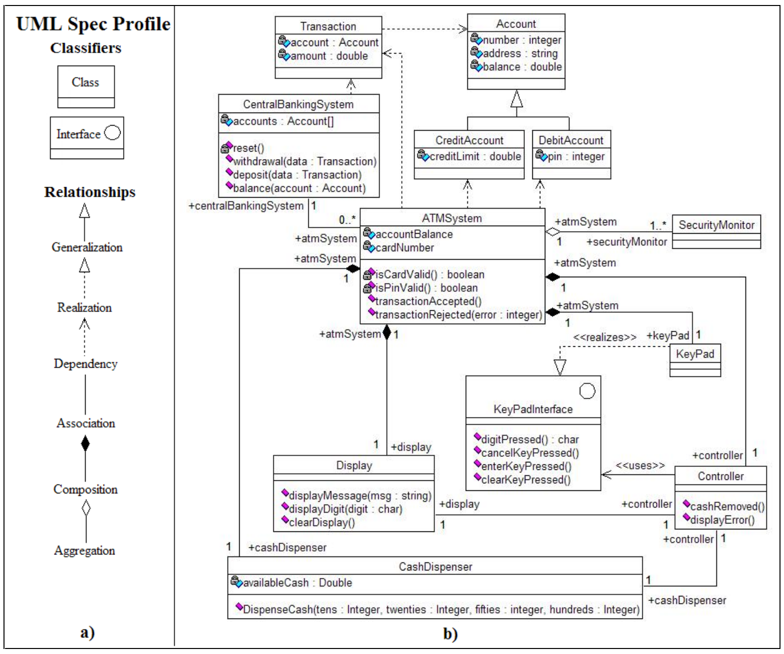

Figure 1. a) UML Spec profile model elements b) ATM UML Spec model example

2.2 UML RRRT 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. The UML RRRT Design profile contains the set of concepts used in the RRRT tool.

Figure 3 a) illustrates the set of UML model elements contained in the UML RRRT Design profile. This set includes two types of model elements:

1) Classifiers:
   - Capsule: Active classes that own its thread of control and communicate through ports. Capsule’s behavior is defined by means of a hierarchical state machine (statechart).
   - Data Class: Passive class that is used as data holder.
   - Protocol: Defines the set of messages that can be sent (outgoing messages) or received (incoming messages) through instances of the protocol class (i.e. port)
   - Port: Instance of a protocol class that allow a capsule to communicate with other capsules or the environment. Ports are owned by Capsules and represent

the only means of communication from a Capsule with other Capsule or with the environment.

2) Relationships: Generalization, realization, dependency, association, composition and aggregation.

The *ATMSystem* of Figure 3 b) is defined using the UML RRRT Design profile. It defines the relationships between the different components that constitute an ATM system. The diagram shows interaction at two different level of abstraction: at a high level it shows interaction between subsystems (*CentralBankingSystem* and *ATMSystem*); at a lower level it shows interaction between the components that constitute the *ATMSystem*. Three types of Classifiers are shown: Data Classes (*Transaction*, *Account*, *CreditAccount*, *DebitAccount* and *DispenseCash_DataClass*), Capsules (*CentralBankingSystem*, *ATMSystem*, *SecurityMonitor*, *UserInterface*, *Display*, *CardReader*, *KeyPad* and *Controller*) and Protocol Classes (*CentralBankingSystemToATMSystem*, *KeyPadToController*, *ControllerToDisplay* and *ControllerToCashDispenser*).

An alternative RRRT view of the conventional UML class diagram of Figure 3 is the capsule diagram of Figure 2. Capsule diagram is the main design diagram used in RRRT. In Figure 3, we see different capsules (identifiable by the icon) defined as the main components in our design. Capsules are represented as rectangles in Figure 2 with the name of the component placed inside of it. Composition relationships from capsules to other capsules in Figure 3 are represented by placing capsules inside other capsules (i.e. Controller, UserInterface and CashDispenser inside CentralBankingSystem) in Figure 2. Composition relationships from capsules to protocol classes (identifiable by the icon) from Figure 3 are represented using small squares placed on the boundaries of the rectangles of Figure 2. A black square defines that the direction of messages that can be sent/received is similar to the direction of the messages (i.e. outgoing/incoming) defined in the protocol class. A white square defines an opposite direction of the messages defined in the protocol class (i.e. outgoing messages defined in the protocol class become incoming messages in the port definition. Incoming messages defined in the protocol class become outgoing messages in the port definition). In RRRT, a black square is known as to play a base role, while a white square is known to play a conjugate role. A connector represented by a solid line bind the ports associated to the capsules.

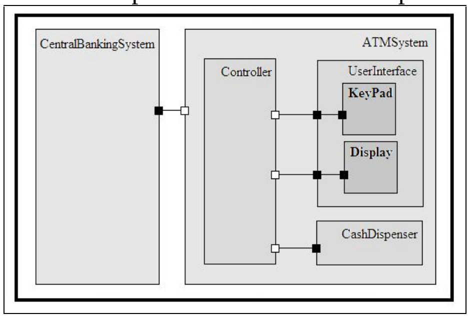

Figure 2. ATM UML RRRT Design capsule diagram

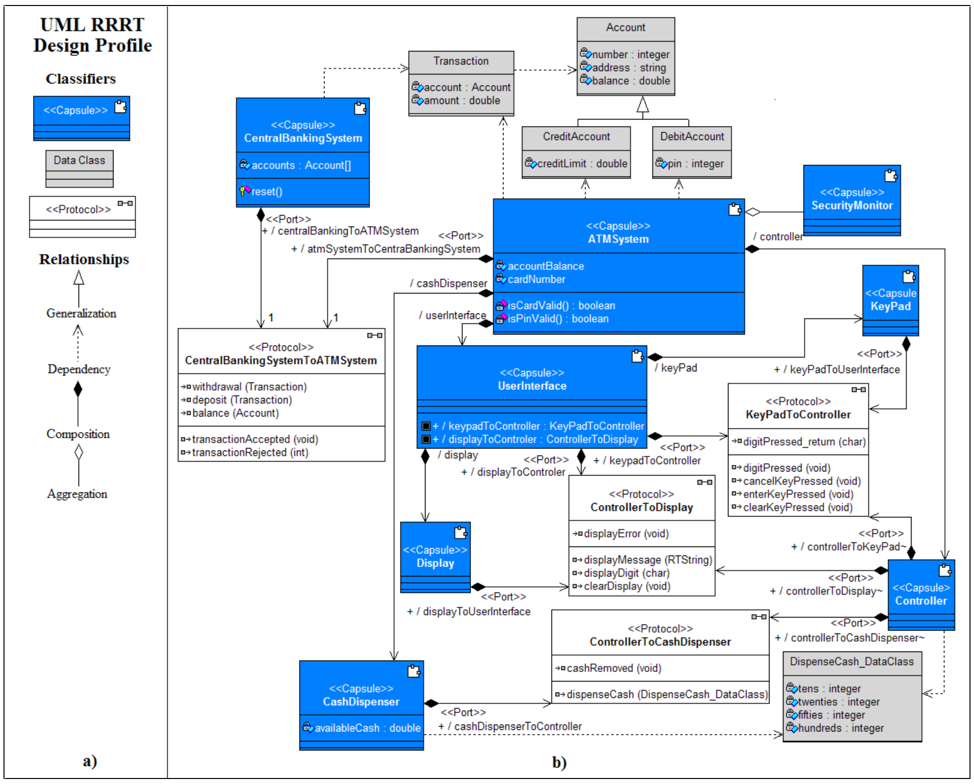

Figure 3. UML RRRT Design Profile. a) Model elements b) ATM model example

In section 4, we make use of the UML Spec and UML RRRT Design profiles to define mechanisms for model mapping.

3. Model Mapping

In *Model Mapping*, we define the general concept of *Mapping* and a set of more specific Mappings for different types of model elements. We use individual packages for the definition of Mapping and the more specific types of Mapping. A Model Mapping is composed of individual mappings. Mappings associate elements on the source model with elements on the target model. In the association, we embed instructions to execute the mapping and, if needed, we add constraint to ensure that the mapping will be applied under specific conditions.

The mapping approach discussed in this paper extends the CWM Transformation Metamodel used for data transformation. The CWM Transformation Metamodel defines mapping structures to map classifiers and features from a source data format into classifiers and features of a target data format. To extend data transformation into model transformation we introduce mapping structures that allow mapping different model elements other than classifiers and features.

3.1 Overview

The *Model Mapping* package contains a set of packages that defines the constructs for mapping models.

The *Model Element* package defines the basic concept of Mapping.

The *Classifier Mapping* package provides mechanisms for specifying structures (as extensions of Mapping) in which the source and target associated elements of Mapping are classifiers.

The *ClassifierFeature Mapping* package provides mechanisms for specifying structures (as extensions of Mapping) in which the source associated elements of Mapping is a classifier and the target associated element of Mapping is a feature.

The *Relationship Mapping* package provides mechanisms for specifying structures (as extensions of Mapping) in which the source and target associated elements of Mapping are relationships.

The *Feature Mapping* package provides mechanisms for specifying structures (as extensions of Mapping) in which the source and target associated elements of Mapping are features.

The *Model Transformation* package provides mechanisms for specifying structures (as extensions of Mapping) to associate more than one Mapping to map model elements that are composed of other model elements.

The *Transformation Map* package provides mechanisms for specifying structures with composite Mappings to map model elements that are composed of other model elements.

The *Transformation Model Element Mapping* package extends Mapping (from Model Element Mapping) with a composite relationship to self.

3.2 Abstract Syntax

Figure 4 shows the dependencies of the Model Mapping packages.

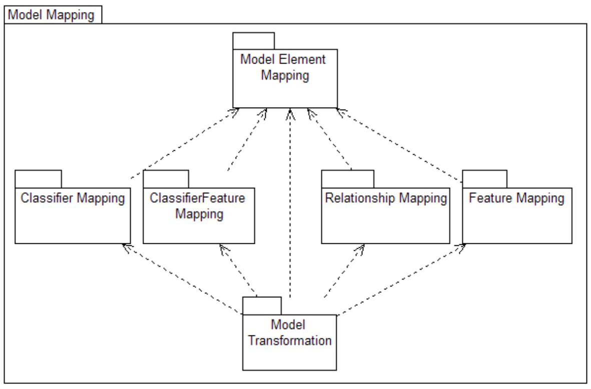

Figure 4 . Model Mapping packages

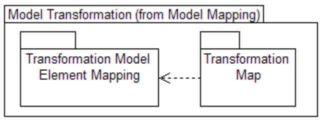
Model Transformation (from Model Mapping)
Transformation Model Element Mapping
Transformation Map

Figure 5. Model transformation packages

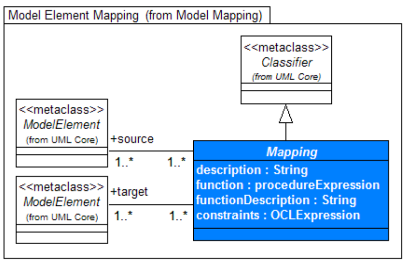
Model Element Mapping (from Model Mapping)
<<metaclass>> Classifier (from UML Core)
<<metaclass>> ModelElement (from UML Core)
+source
1..*
1..*
<<metaclass>> ModelElement (from UML Core)
+target
1..*
1..*
Mapping
description : String
function : procedureExpression
functionDescription : String
constraints : OCLExpression

Figure 6. Mapping definition (from Model Element Mapping)

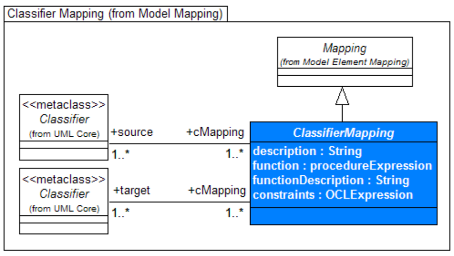
Classifier Mapping (from Model Mapping)
Mapping (from Model Element Mapping)
<<metaclass>> Classifier (from UML Core)
+source
+cMapping
1..*
1..*
<<metaclass>> Classifier (from UML Core)
+target
+cMapping
1..*
1..*
ClassifierMapping
description : String
function : procedureExpression
functionDescription : String
constraints : OCLExpression

Figure 7. ClassifierMapping

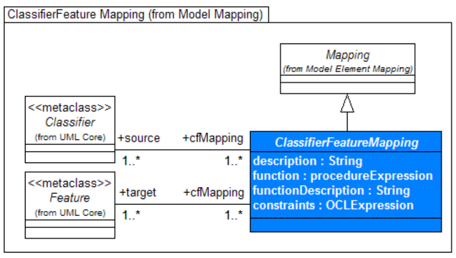
ClassifierFeature Mapping (from Model Mapping)
Mapping (from Model Element Mapping)
<<metaclass>> Classifier (from UML Core)
+source
+cfMapping
1..*
1..*
<<metaclass>> Feature (from UML Core)
+target
+cfMapping
1..*
1..*
ClassifierFeatureMapping
description : String
function : procedureExpression
functionDescription : String
constraints : OCLExpression

Figure 8. ClassifierFeatureMapping

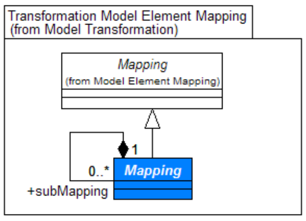
Transformation Model Element Mapping
(from Model Transformation)
Mapping
(from Model Element Mapping)
1
0..*
Mapping
+subMapping

Figure 9. Transformation Model Element Mapping

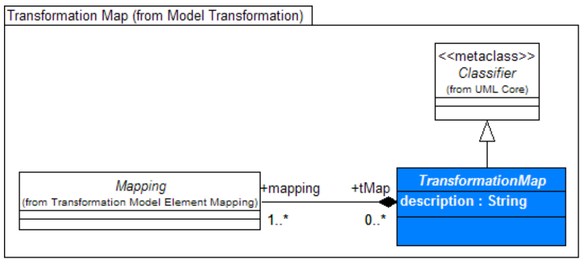

Figure 10. Transformation Map

3.3 Class Descriptions

Mapping (from Model Element Mapping) is the basic structure for model mapping. Mapping defines properties that provide information related to a mapping. Mapping is an abstract specialization of classifier with associated *ModelElement* that play the *source* and *target* model elements of the mapping. Tagged values are also embedded as attributes of the Mapping stereotype. The attributes hold information needed to execute the mapping.

Tagged values in the Mapping definition include the following attributes: *description*, *function*, *functionDescription* and *constraints*. The attribute *description* provides a text description of the objective of the mapping. The attributes *function* and *functionDescription* contain information on how to execute the mapping. The *constraints* attribute defines specific conditions (if needed) required to execute the mapping. Examples of values used for the attributes are provided in Figure 13.

ClassifierMapping (from Model Element Mapping) extends Mapping (from Model Element Mapping). ClassifierMapping defines that the source and target model elements associated to ClassifierMapping are classifiers.

ClassifierFeatureMapping (from Model Element Mapping) extends Mapping (from Model Element Mapping). ClassifierFeatureMapping defines that the source and target model elements associated with ClassifierFeatureMapping are classifiers and features respectively.

RelationshipMapping (from Model Element Mapping) extends Mapping (from Model Element Mapping). RelationshipMapping defines that the source and target model elements associated with RelationshipMapping are relationships. RelationshipMapping is similar to ClassifierMapping in its definition using relationships instead of classifiers. Class diagram is not shown in the abstract syntax in section 3.2

FeatureMapping (from Model Element Mapping) extends Mapping (from Model Element Mapping). FeatureMapping defines that the source and target model elements associated with FeatureMapping are features. FeatureMapping is similar to ClassifierMapping in its definition using features instead of classifiers. Class diagram is not shown in the abstract syntax in section 3.2

Mapping (from Transformation Model Element Mapping) extends Mapping (from Model Element Mapping) by defining a composition relationship to self. The composition relationship allows the mapping of model elements that are composed of other model elements. Mapping is used to create TransformationMap (from Transformation Map).

TransformationMap (from Transformation Map) is defined as an extension of a classifier with a composition relationship to Mapping (from Transformation Model Element Mapping). TransformationMap is used to map model elements that are composed of other model elements (i.e. classes with operations). The composite Mapping (from Transformation Model Element Mapping) of TransformationMap maps the model element (*owner*) composed of other model elements (*owned*). Further composite Mapping map the owned model elements of the owner model element.

4. UML Spec to UML RRRT Design Mapping

In this section, we define the *UML Spec to UML RRRT Design Mapping* that allows mapping elements of the UML Spec profile into elements of the UML RRRT Design profile. The UML Spec to UML RRRT Design Mapping and UML Spec to UML RRRT Design Mapping packages are graphically described in Figure 11 and Figure 12**.**

4.1 Overview

This section describes the different elements that compose the UML Spec to UML RRRT Design Mapping package. Elements that were described in previous sections of this paper (Model mapping in section 3, UML Spec profile in section 2.1, and UML RRRT Design profile in section 2.2) are not further discussed in this section.

The UML Spec to UML RRRT Design Mapping package provides mechanisms for specifying structures for profile-to-profile mapping. The UML Spec to UML RRRT Design Mapping packages (Profile Specific Model Element mapping and Profile Specific Transformation Map) specialize structures from the Model Mapping package. These packages define that the source model elements of a mapping are defined in the UML Spec profile and the target model elements of a mapping are defined the UML RRRT Design profile. We express this definition by stereotyping the dependency relationships from the UML Spec to UML RRRT Design Mapping package to the UML Spec profile Package and the UML RRRT Design profile package.

The Profile Specific Model Element Mapping package provides mechanisms for specifying structures (as instances of Mapping specializations) in which the source and target associated elements are defined in the source and target profiles respectively.

The Profile Specific Transformation Map subpackage provides mechanisms for specifying structures of Mapping specializations in which the associated elements of Mapping are defined in the source and target profiles.

4.2 Abstract Syntax

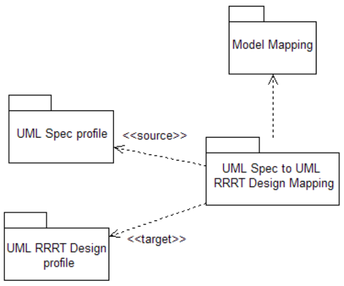

Figure 11.UML Spec to UML RRRT Design Mapping

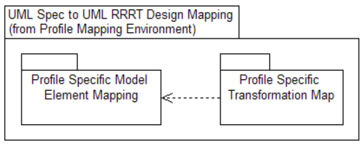
UML Spec to UML RRRT Design Mapping (from Profile Mapping Environment)
Profile Specific Model Element Mapping
Profile Specific Transformation Map

Figure 12. UML Spec to UML RRRT Design Mapping packages

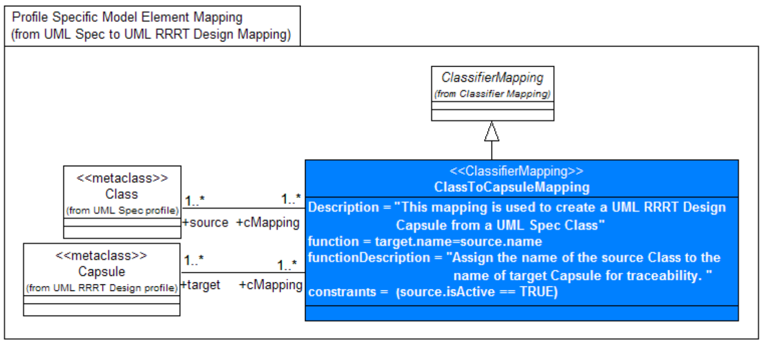
Profile Specific Model Element Mapping (from UML Spec to UML RRRT Design Mapping)
ClassifierMapping (from Classifier Mapping)
<<metaclass>> Class (from UML Spec profile)
<<metaclass>> Capsule (from UML RRRT Design profile)
1..*
1..*
+source
+cMapping
1..*
1..*
+target
+cMapping
<<ClassifierMapping>> ClassToCapsuleMapping
Description = "This mapping is used to create a UML RRRT Design Capsule from a UML Spec Class"
function = target.name=source.name
functionDescription = "Assign the name of the source Class to the name of target Capsule for traceability. "
constraints = {source.isActive == TRUE}

Figure 13. ClassToCapsuleMapping

The ClassMap (from Profile Specific Transformation Map) definition, part of the abstract syntax from section 4.2 is shown in Figure 14 below.

4.3 Class Descriptions

ClassToCapsuleMapping (from Profile Specific Model Element Mapping) specializes ClassifierMapping. ClassToCapsuleMapping is associated with a class as the source model element, and with a Capsule as the target model element. ClassToCapsuleMapping also defines values for its attributes. Among them, we highlight the *function* attribute that ensures traceability by using the class name as the capsule name for traceability purposes. The *constraints* attribute ensures that only active classes from the UML Spec profile are mapped into capsules of the UML RRRT Design profile.

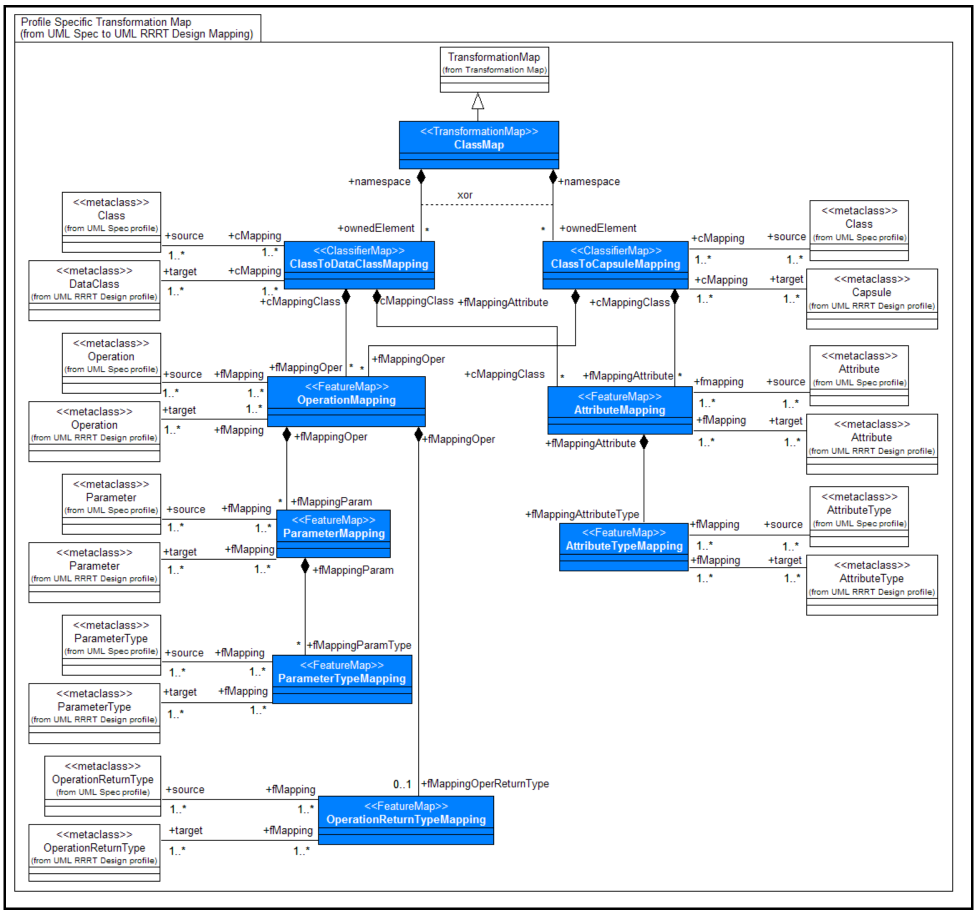
Profile Specific Transformation Map
(from UML Spec to UML RRRT Design Mapping)
TransformationMap (from Transformation Map)
<<TransformationMap>> ClassMap
+namespace
xor
+ownedElement
<<ClassifierMap>> ClassToDataClassMapping
<<ClassifierMap>> ClassToCapsuleMapping
<<FeatureMap>> OperationMapping
<<FeatureMap>> AttributeMapping
<<FeatureMap>> ParameterMapping
<<FeatureMap>> AttributeTypeMapping
<<FeatureMap>> ParameterTypeMapping
<<FeatureMap>> OperationReturnTypeMapping
<<metaclass>> Class (from UML Spec profile)
<<metaclass>> DataClass (from UML RRRT Design profile)
<<metaclass>> Capsule (from UML RRRT Design profile)
<<metaclass>> Operation (from UML Spec profile)
<<metaclass>> Operation (from UML RRRT Design profile)
<<metaclass>> Attribute (from UML Spec profile)
<<metaclass>> Attribute (from UML RRRT Design profile)
<<metaclass>> Parameter (from UML Spec profile)
<<metaclass>> Parameter (from UML RRRT Design profile)
<<metaclass>> AttributeType (from UML Spec profile)
<<metaclass>> AttributeType (from UML RRRT Design profile)
<<metaclass>> ParameterType (from UML Spec profile)
<<metaclass>> ParameterType (from UML RRRT Design profile)
<<metaclass>> OperationReturnType (from UML Spec profile)
<<metaclass>> OperationReturnType (from UML RRRT Design profile)
+source
+target
+cMapping
+fMapping
+cMappingClass
+fMappingOper
+fMappingAttribute
+fMappingParam
+fMappingAttributeType
+fMappingParamType
+fMappingOperReturnType
1..*
0..1
*

Figure 14. ClassMap (from Profile Specific Transformation Map) definition.

ClassMap (from Profile Specific Transformation Map) is defined to map a UML Spec class into a UML RRRT Design data class or capsule. ClassMap is defined in terms of composition relationships to *ClassToDataClassMapping* and *ClassToCapsuleMapping*. A constraint associated with the two composition relationships from ClassMap prevents from applying both mappings to the same class. More specifically, ClassMap will either use *ClassToDataClassMapping* or *ClassToCapsuleMapping* for the transformation, but not both. The selection on which mapping is to be used is based on the constraint defined for each mapping. In this case, *ClassToDataClassMapping* holds a constraint source.isActive==FALSE while *ClassToCapsuleMapping* will be chosen when source.isActive==TRUE. The result on the execution on the selected mapping will create a classifier on the UML RRRT Design model. Both mappings are specialization of

ClassifierMap (source and target elements are classifiers). Once the classifier is created on the UML RRRT Design model, we map the operations and attributes of the class as operations and attributes of the data class or capsule of the UML RRRT Design model.

Due to the lack of space we refer the reader to [7] for definition of ClassToDataClassMapping, OperationReturnTypeMapping, AttributeTypeMapping, OperationMapping, ParameterMapping, ParameterTypeMapping, AttributeMapping and.

5. UML Spec to UML RRRT Design ATM Example.

In this section, we explain how the ATM UML Spec model from Figure 1 b) was mapped into the ATM UML RRRT Design model of Figure 3 b).

In the example, we first specialize different types of Mappings (from packages of the Model Mapping package) to associate profile specific model elements from the source UML Spec and the target UML RRRT Design profiles. The definition of this profile specific set of Mappings is performed in three main steps:

1) Analysis of the concepts defined in the UML Spec and UML RRRT Design profiles.
2) Definition of conceptual relationships between the elements of the UML Spec and UML RRRT Design profiles.
3) Formalize the relationships in a well-defined Profile Specific Mapping package. Embed the element's association, traceable naming and constrain definition (if needed) in the definition o a set of mapping stereotypes.

Table 1 illustrates the result of steps 1 and 2 applied to some of the elements in the UML Spec and UML RRRT Design profiles. Direct associations are the basis for the definition of the mappings. A direct association may be further refined into a more complex type of association. When no direct association is established, a mapping mechanism needs to be defined instead. Elements from the target profile with no direct association to an element of the source profile may be used in such mechanisms (or in refinements of direct associations). The goal is to define mappings for all the elements used (defined) in the source and target profiles. Direct associations are established with the Class classifier, with the Attribute and Operation features, and with several relationships. Realization and Association relationships are elements from the source profile with no direct association to the target profile. An example of how a mapping mechanism can be defined for an association relationship is described below. From the target profile, we see elements with no direct association with an element of the source profile: Capsule and Port that specializes the metaclass Class, and Protocol, which specializes the metaclass Collaboration.

In Figure 13 we define a profile specific mapping. *ClassToCapsuleMapping* specializes ClassifierMapping. The diagram defines associations between specializations of the Classifier metaclass from the UML Spec and UML RRRT Design profiles. ClassToCapsuleMapping also defines values for its attributes. Among them, we highlight the *function* attribute that ensures traceability by using the class name as the capsule name for traceability purposes. The *constraints* attribute ensures that only active classes from the UML Spec profile are mapped into capsules of the UML RRRT Design profile.

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

	Model Element	Metaclass	UML Spec	UML RRRT Design
⊟	Class (UML Spec) Data Class (UML RRRT)	Classifier	✓	✓
⧉	<<Capsule>>	Classifier		✓
▫-▫	<<Protocol Class>>	Classifier		✓
▣	<<Port>>	Classifier		✓
	Attribute	Feature	✓	✓
	Operation	Feature	✓	✓
↑	Generalization	Relationship	✓	✓
⇡	Realization	Relationship	✓	
◆	Composition	Relationship	✓	✓
⇣	Dependency	Relationship	✓	✓
◇	Aggregation	Relationship	✓	✓
↔	Association	Relationship	✓	

At the classifier level, we used *ClassMap* Transformation Map from Figure 14 to map UML Spec passive classes (*Transaction*, *Account*, *CreditAccount* and *DebitAccount*) into UML RRRT Design data classes (*Transaction*, *Account*, *CreditAccount* and *DebitAccount*). Using the same ClassMap TransformationMap we also mapped UML Spec active classes (*CentralBankingSystem*, *ATMSystem*, *SecurityMonitor*, *Display*, *CardReader*, *KeyPad* and *Controller*) into UML RRRT Design capsules (*CentralBankingSystem*, *ATMSystem*, *SecurityMonitor*, *Display*, *CardReader*, *KeyPad* and *Controller*). The *KeyPad* Interface shown in Figure 1 b) was mapped into a protocol class (*KeyPadToController*) using the *InterfaceToProtocolClassMapping* (not described in this paper). Features of classifiers were mapped using mappings also defined in Figure 14. In most of the Mappings, we use the *function* property to ensure traceability by assigning the same name of the source model element to the name of the target model element.

To map interaction between classes we mapped associations between active classes (*CentralBankingSystem* and *ATMSystem*, *Controller* and *Display*, and *Controller* and *CashDispenser*) into a communication mechanism using protocol classes and port definition. In the UML RRRT Design profile, capsule communication is realized through the use of ports and bindings, which connect ports of communicating capsules. The realization of this communication mechanism includes protocol class definition with the set of incoming/outgoing messages, and composition relationships from the capsules to the protocol class for port creation. A connector binding the ports of the composed protocol class establishes the required communication channel for capsule interaction. The same communication approach is used to address the interaction between the *Controller* and *KeyPad* using the *KeyPadInterface*. We use *AssociationMap* (not described in this paper) to map associations from the UML Spec profile into the communication mechanism described above. In applying *AssociationMap*, the *DispenseCash_DataClass* shown at the bottom right of Figure 3 b) was created. The new data class is direct result of applying the *OperationToDataClassMapping* (not described in this paper) which is included in the *AssociationMap* TransformationMap definition. The *OperationToDataClassMapping* defines the mapping procedure to map operations with more than two parameters into a new data class to be attached with the message.

The *UserInterface* capsule from Figure 3 is the result of a design decision to integrate into a single component the functionality required for *KeyPad* and *Display* classes

Due to lack of space, we refer the reader to [7] for more information on this mapping example.

6. Conclusion

This paper describes the approach we use for Profile definition and 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. The concept of Model Mapping is introduced. Constructs and mechanisms for model mappings are defined. We illustrate how we use such constructs for the creation of Profile Specific Mapping models. This mapping allows making the transition between a UML spec and a UML RRRT Design in a systematic and traceable manner.

7. 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. OMG Specification. *UML 2.0 Superstructure revised submission.* http://www.omg.org/techprocess/meetings/schedule/UML_2.0_Superstructure_RFP.
4. B. Selic, J. Rumbaugh, *Using UML to Model Complex Real- Time Systems*, Rational whitepaper: http://www.rational.com/media/whitepapers/umlrt.pdf
5. David S. Frankel. *Applying MDA to Enterprise Computing.* OMG Press 2003.
6. Grady Booch et al. *The Unified Modeling Language User Guide.* Addison Wesley. 1998
7. OMG Specification. Common Warehouse Metamodel (CWM) Specification. www.omg.org.
8. Juan Pablo Zamora Zapata. From Specification level to Design level: A Model Driven Architecture (MDA) Transformation Model. Ph. D. Thesis, Carleton University, Ottawa, Ontario, Canada. In progress.