A Framework to Incorporate Non-Functional Requirements into UML Models

Subrina Anjum Tonu, Ladan Tahvildari

Software Technologies Applied Research Lab

Dept. of Electrical & Computer Engineering

{subrina,ltahvild}@swen.uwaterloo.ca

Abstract

Despite the fact that Non-Functional Requirements (NFRs) are very difficult to achieve and at the same time are expensive to deal with, a few research works have focused on them as first class requirements in a development process. We propose a framework to incorporate NFRs, as reusable components, into standard UML notations. Such a framework can also integrate those reusable NFRs with the extracted UML representations of legacy systems during the reverse engineering process. This novel research work uses standard XMI representation of UML models without proposing any extension to it.

Problem Statement

- Production of a high quality software requires implementation of all functional and non-functional requirements starting from the design phase to the end of the software life cycle.
- All these requirements are changing during the maintenance phase of any software system.
- Existing reverse Engineering process can extract architectural design of legacy system as UML model.
- UML tools also exist to automatically generate deployable source code from UML model specifications.
- It is necessary to have an environment to attach the new FRs and NFRs to the target system.
- In current practice, the join-point (where the NFR touches the target model) is defined as a part of the NFR itself. As a result, there is very little chance to reuse this NFR in other software design.

Research Focus

- *Design and develop a methodology for*

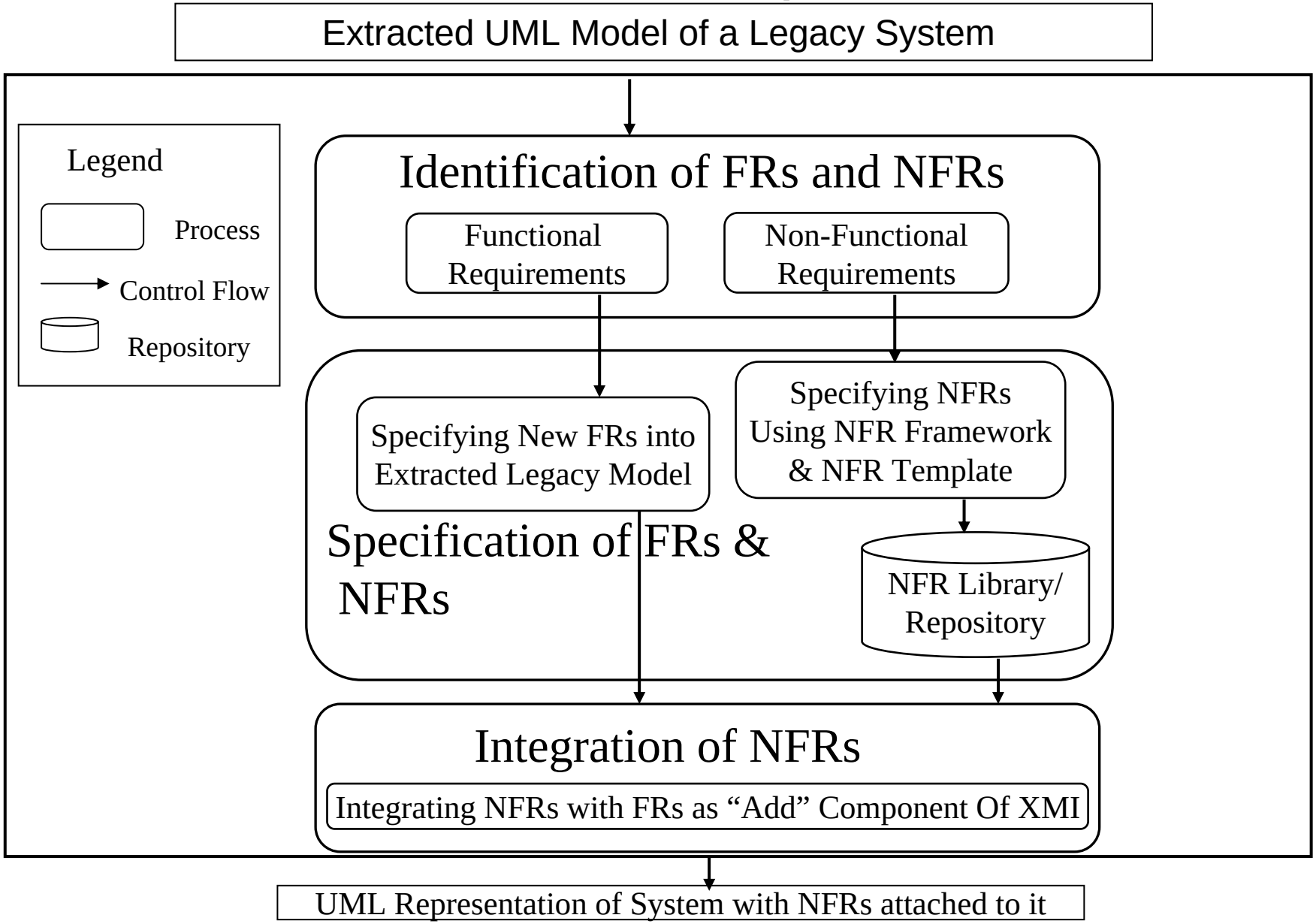
  *"**Making Reusable NFRs Library/Repository with Dynamic Parameterization of their join-points with the target model**"*

- *Need a comprehensive framework to*
  - Design Reusable NFRs with standard UML notations
  - Add NFRs into the extracted UML model of legacy system from the source code.

Goals and Issues

- *Provide a high level notation to design NFRs using UML models*
- *Exploit the synergy between*
  - Requirements analysis (both Functional and Non-functional)
  - Software re-engineering
- *Preserve the original system functionality.*
- *Add new NFRs into the original system to improve its quality*

A Framework to Incorporate NFRs



Framework Steps

- **Identification of FRs and NFRs from**
  - Extracted UML model of a legacy system
  - Design documents
  - New user's requirements
- **Specification of FRs and NFRs**
  - Specify FRs with UML diagram
  - Specify NFRs using NFR Framework and our proposed
  - High level notations for building template of NFRs
- **Integration of NFRs**
  - Integrate NFRs as "Add" component of XMI

Proposed High Level Notation

- Providing commands for creating each type of UML entity.
- NFR template is a set of ordered commands for creating UML representations.
- The template does not contain any hard-coded join-points of NFR with the target model.
- These commands are executed during the weaving of NFRs with the target model.
- The necessary parameters are supplied during run time.

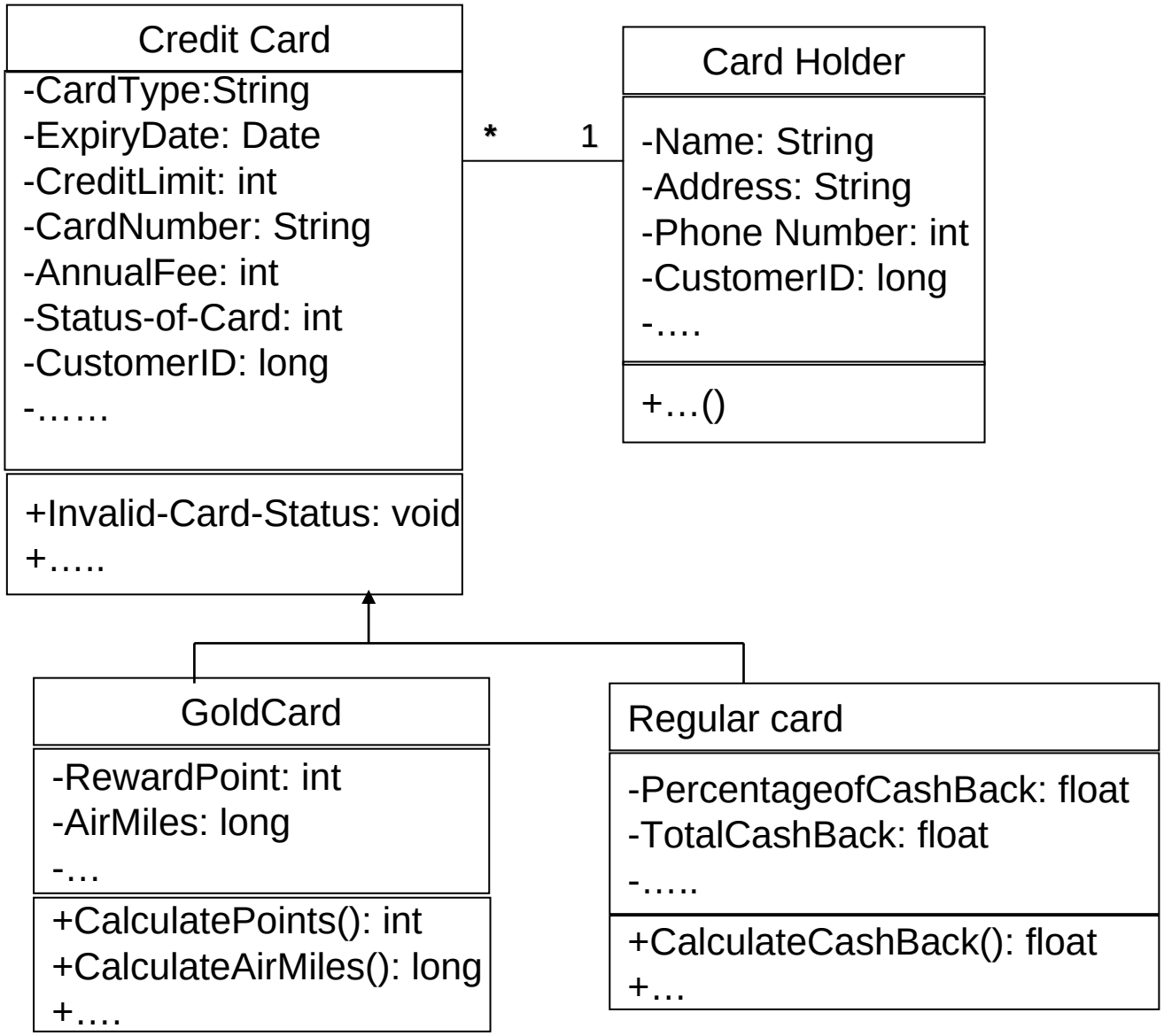
NFR Library/Repository

- Building the NFR Library/Repository with those NFR templates.
- The Library/Repository can be evolved during the time.

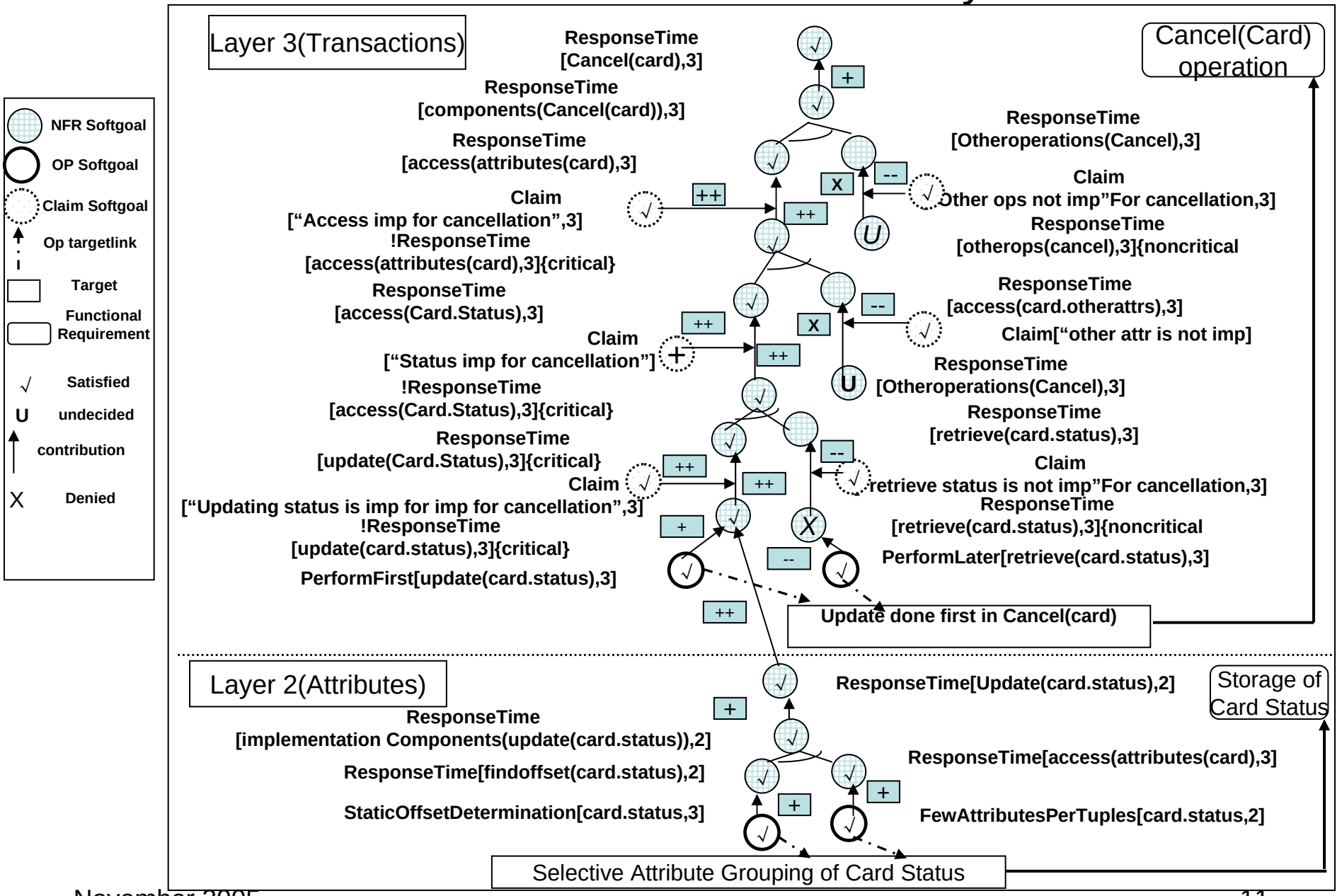
Case Study – Credit Card System

- FRs
  - maintaining information on sales, card holders and merchants
  - Transactions are authorized, and accounts are updated.
  - Stolen cards are cancelled.
- NFRs
  - Fast Response time for card cancellation method

Class Diagram of Credit Card System



SIG Performance for Credit Card System



Process for Incorporating NFR FastResponseTime

- **Step 1:**Accepting the join-point of the NFR ``Fast Response Time''. Here, method Invalid-Card-Status().
- **Step 2:**Selecting other attributes to move to the separate class for card status (here attributes Card-Number and Status-of-card).
- **Step 3:**Creating a new class (say, FastResponseTime) and building an association with the parent class.
- **Step 4:**Inserting those attributes defined in Step 2 and the method defined in Step 1, into this new class.
- **Step 5:**Deleting the attribute Status-of-card from the original class.

NFR Template for NFR

Steps	Commands
0:	InputParameters: Method(Single)<param0>
1:	AttributeList<result0> := getAttributesFromParentClass(<param0>)
2:	AttributeList<result1> := getSelectedInput(“Attributes to Move”, <result0>)
3:	AttributeList<result2> := getSelectedInput(“Primary Keys”, <result0)
4:	ClassNode<result3> := createNewClass(“FastResponseTime”)
5:	<result3> := insertMultipleAttribute(<result3>, <result2>)
6:	<result3> := insertMultipleAttribute(<result3>, <result1>)
7:	<result3> := insertMethod(<result3>, <param0>)
8:	deleteMethodFromParentClass(<param0>)
9:	deleteAttributeFromParentClass(<result1>)
10:	ClassNode<result4> := getParentClass(<param0>)
11:	createAssociation(<result3>, <result4>,1,1)
12:	Output: (Step 7(result), Step8 8(Operation), Step 9 (Operation), Step11(Operation))

Java Class of NFR Template

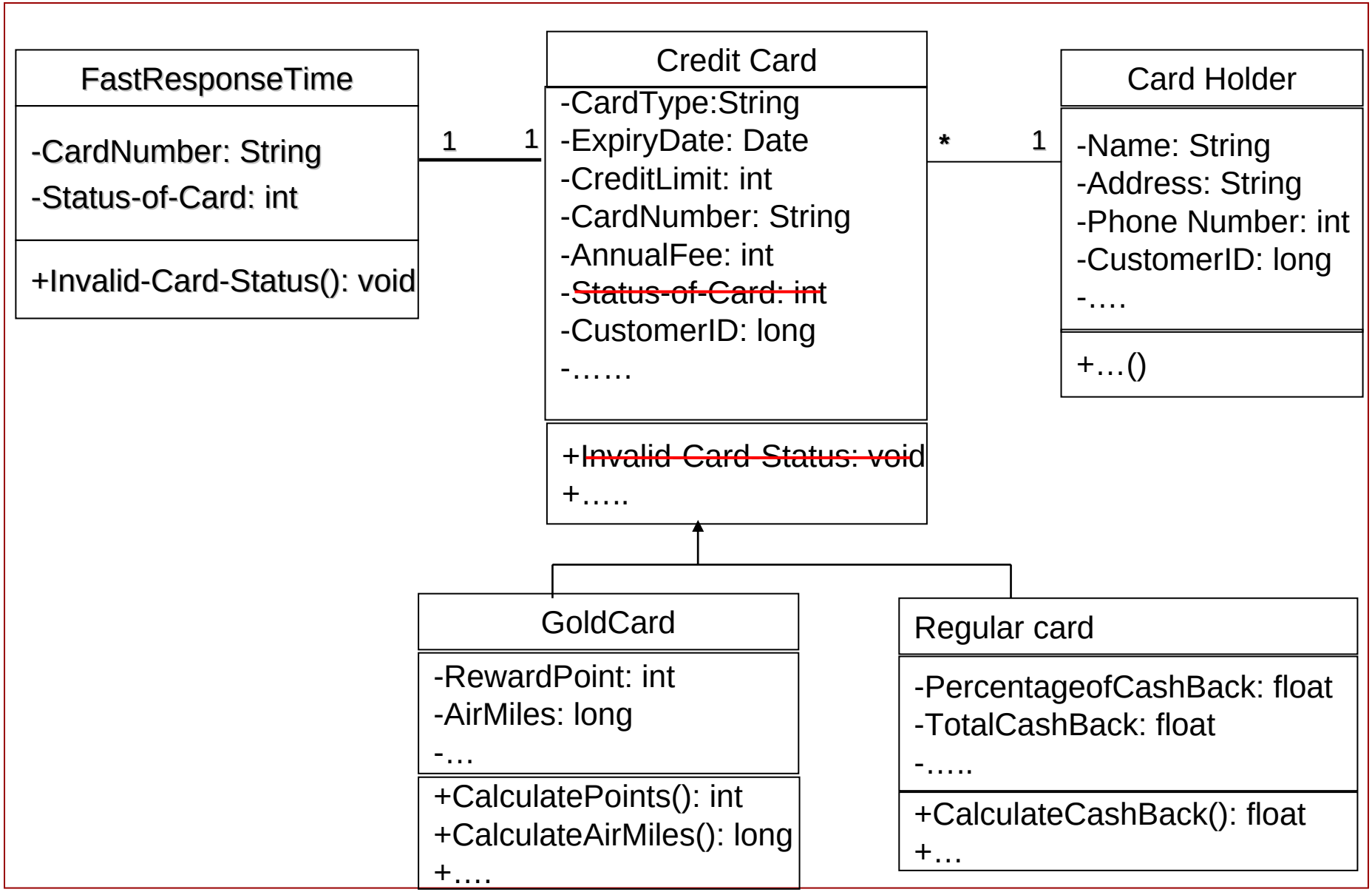
```
public class FastResponseTime{

     public FastResponseTime(){

    }
   void constructTargetSystem(){

        expectParameter(Method);
        MethodName param0 = getParameter();
        AttributeList result0 = getAttributesFromParentClass(param0);
        AttributeList result1 = getSelectedInput("Attributes to Move", result0);
        AttributeListresult2 = getSelectedInput("Primary Key", result0);
        ClassElement result3 = createNewClass("FastResponseTime");
        result3 = insertMultipleAttribute(result3,result1);
        result3 = insertMultipleAttribute(result3,result2);
        result3 = insertMethod(result3,param0);
        deleteMethodFromParentClass(param0);
        deleteAttributeFromParentClass(result1);
        ClassElement result4 = getParentClass(param0);
         createAssociation(result3,result4,1,1);
      }
}
```

Adding NFR FastResponseTime



Conclusions

- *Propose a framework to integrate NFRs using UML models.*
- *Improve the quality of a legacy system by applying the framework.*

- *Need to accomplish the following steps:*
  - Extract the UML representation of an object-oriented legacy system
  - Identify FRs and NFRs
  - Specify new FRs (UML model) and NFRs (template for UML model)
  - Integrate NFRs with the FRs
  - Get the output as UML models of the legacy system with NFRs attached to it