3. ULUSAL YAZILIM MİMARİSİ KONFERANSI '10

Bilkent Üniversitesi

AKILLI KART YAZILIMLARININ MODEL GÜDÜMLÜ GELİŞTİRİLMESİ

Hidayet Burak SARITAŞ Geylani KARDAŞ

Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü

4 Kasım 2010

İçerik

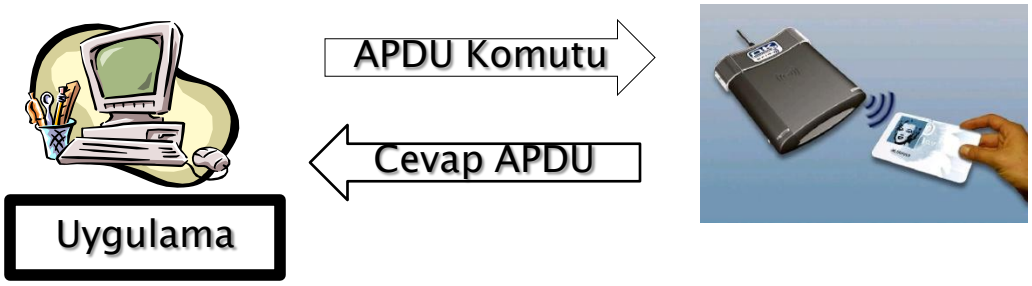
- Akıllı kartlar
- Amaç
- Model Güdümlü Uygulama Geliştirme
- Platform Bağımsız Akıllı Kart Modeli
- Platforma Özgü Modeller
- Akıllı Kart Uygulamalarının Modellenmesi
  - Cüzdan uygulaması
- Modeller arası Dönüşüm (M2M)
  - Platform-Bağımsız Modelden Platforma-Özgü Modellere(M2M)
- PSM (Platforma-Özgü) aşamasında Modelleme
- Akıllı Kart Uygulamaları için Otomatik Kod Üretme
  - Üretilen Kodun İçeriği
- Plan ve Sonuç

Akıllı Kartlar

- Dahili mikro işlemciye sahip plastik kartlar
- Veriyi işleme ve saklama yeteneğine sahip
- ISO/IEC 7816 ile standartlaştırılmış fiziksel ve iletişim altyapısı
- Kullanım alanları:
  - Kredi kartları – Sim kartları
  - Pasaportlar – Ulaşım kartları
  - Sağlık kartları – Kimlik kartları
- Kullanım nedenleri:
  - Güvenlik
  - Taşınabilirlik
  - Otomasyon

İletişim Altyapısı

- APDU (Application Protocol Data Unit)
  - Akıllı kart ile okuyucu arasında tüm iletişim, **uygulama protokolü veri yapısı** olarak adlandırılan APDU paket yapısı ile sağlanır
  - Geliştirilen uygulamalar bu paket yapısını kullanarak, alttaki protokolden bağımsız olarak kart ile iletişim kurabilir.



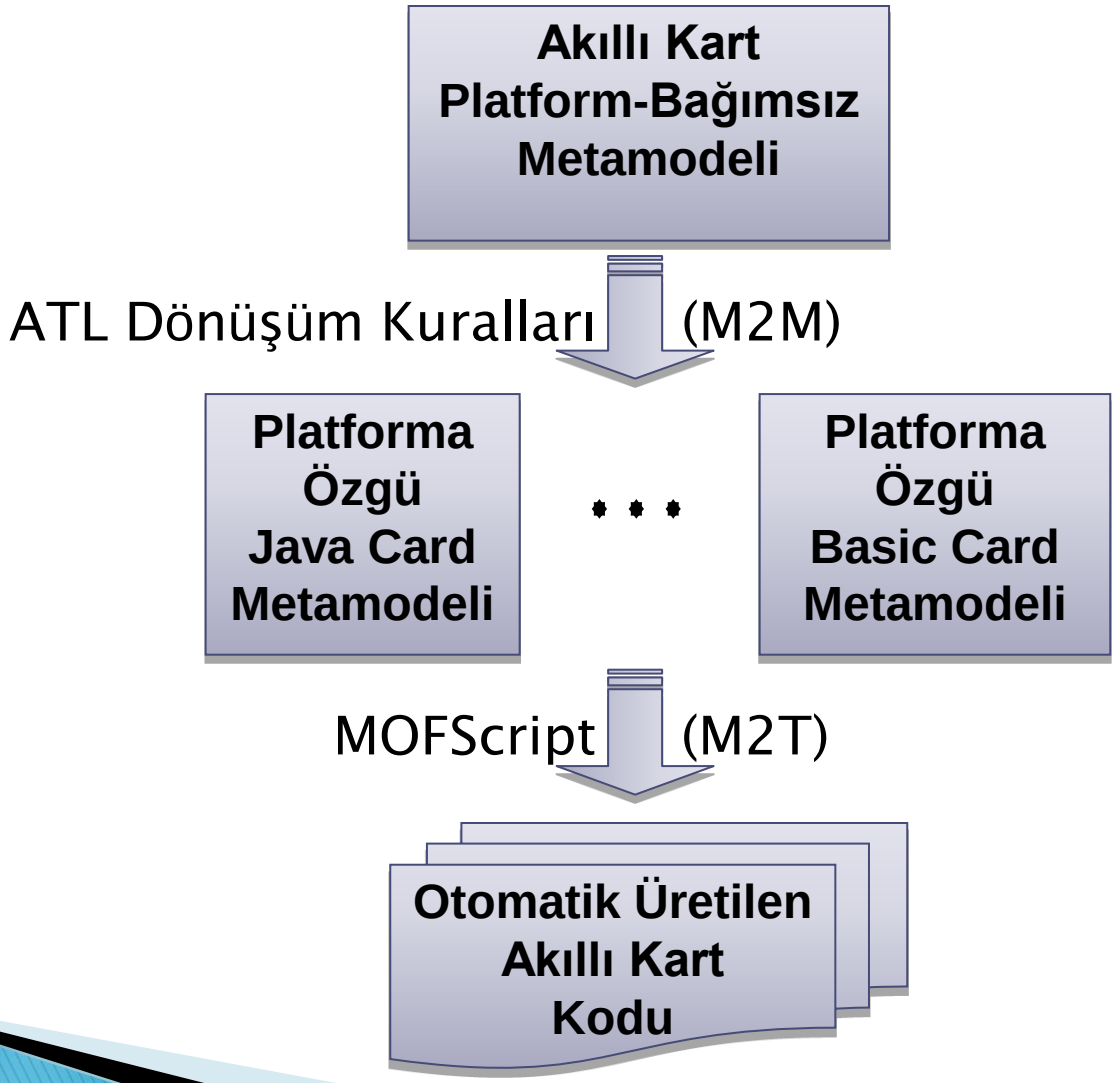
Akıllı Kartlar için Uygulama Geliştirme

- Farklı akıllı kartlar için farklı uygulama geliştirme platformları
  - Java Card – (Java Card Development Environment)
  - Basic Card – (ZC-Basic Language)
- Alt Seviye iletişim yapısı
- Donanımsal sebepler
  - Sınırlı Hafıza

Amaç

- Belli bir dile bağlı kalmaksızın, farklı platformlara, ISO/IEC 7816 standartlarına ve temel programlama gereklerine uyumlu platform bağımsız bir akıllı kart metamodeli oluşturmak
- Metamodelden örnek modeller oluşturabilmek için görsel bir kullanıcı arayüzü
- Farklı akıllı kart platformlarına özgü metamodeller ve görsel bir kullanıcı arayüzü oluşturmak
  - Java Card – Basic Card
- Modeller arası dönüşüm kuralları oluşturmak(M2M)
- Hazırlanan modellerden akıllı kart koduna dönüşüm işlemleri (M2T)

Yaklaşım

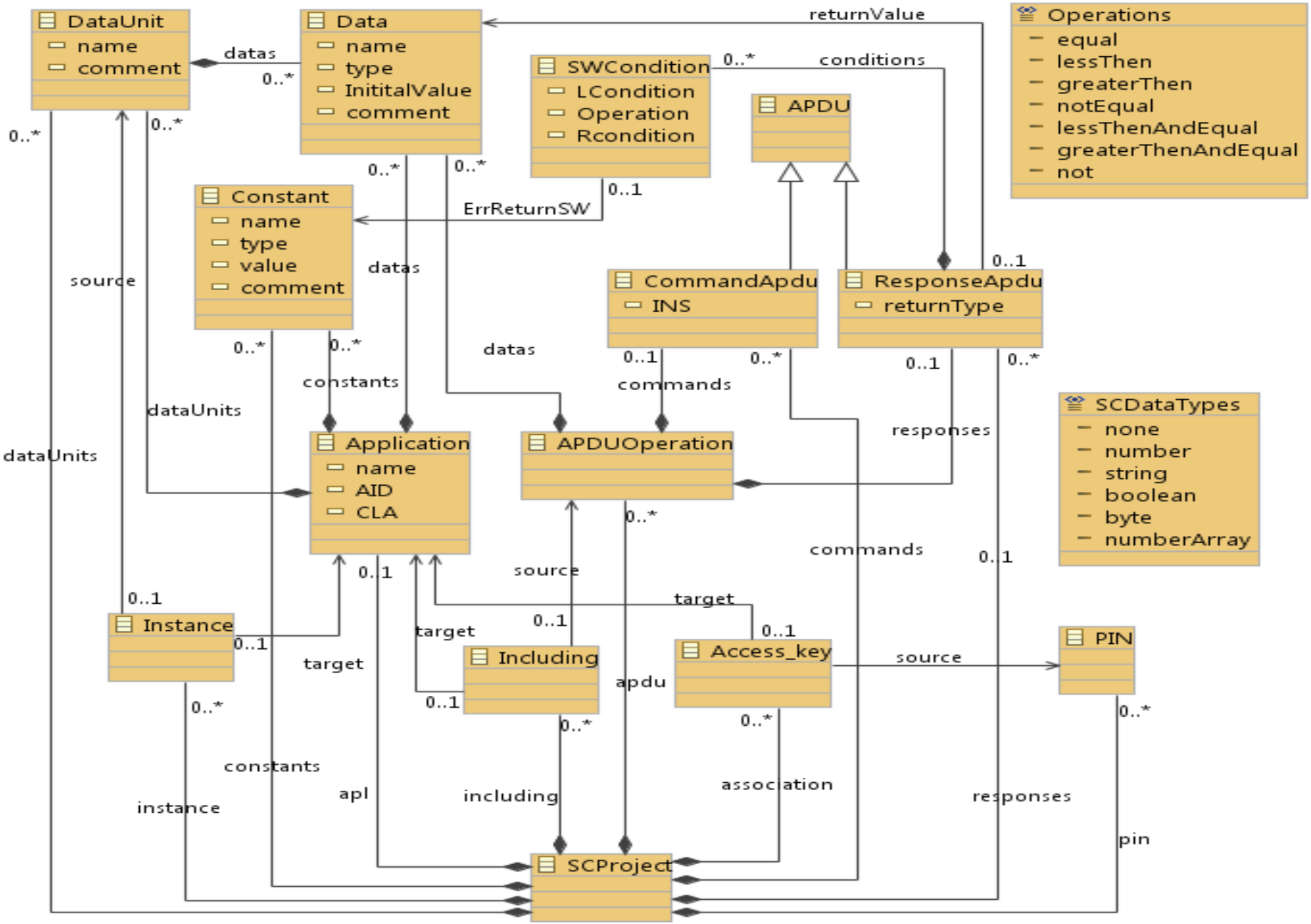

Akıllı Kart Platform-Bağımsız Metamodeli
ATL Dönüşüm Kuralları
(M2M)
Platforma Özgü Java Card Metamodeli
Platforma Özgü Basic Card Metamodeli
MOFScript
(M2T)
Otomatik Üretilen Akıllı Kart Kodu

Model Güdümlü Uygulama Geliştirme

- Model Güdümlü Geliştirme (MDD – Model Driven Development) ile yazılım geliştirme süreci modeller üzerinden tanımlanmaya çalışılmıştır.
- Gerçek anlamda, üretilen modeller, Model Tabanlı Mimari (MDA) bakış açısına göre akıllı kartlar için platform bağımsız model, Java Card ve Basic Card için platforma özgü metamodeller olarak adlandırılabilir
- Geliştirilen Metamodeller Eclipse modelleme ortamı üzerinde Ecore metamodeli kullanılarak tasarlanmıştır
  - Eclipse Modeling Framework (EMF)
    - EMF, yeni bir model geliştirebilme ve bu model için gerçek zamanlı kod üretebilmeyi sağlayan araçlar sunmaktadır

Platform Bağımsız Akıllı Kart Modeli

- Platform Bağımsız Metamodel
  - Belli bir uygulama geliştirme platformuna bağlı kalmadan akıllı kart uygulamaları geliştirebilmek
  - Daha sonra eklenebilecek dile özgü platformlara uyarlanabilecek esnek bir yapı
  - Modellerin kolay anlaşılır ve geliştirilebilir olması



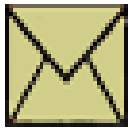
- Hazırlanan Akıllı Kart Metamodeli

Akıllı Kart Metamodel Elemanları

- Metamodelin anahtar öğesi *Application* model elemanıdır
  - Esas olarak bir akıllı kart programını temsil etmektedir
  - Akıllı kart uygulaması geliştirirken bir başlangıç noktası olarak görülmekte ve her akıllı kart örnek modeli içine eklenmektedir
  - Modelleme ortamında tanımlanabilen tüm sabitler, işlemler, ilişkiler ve veri tipleri tümüyle *Application* elemanı ile bağlantılıdır.

Akıllı Kart Metamodel Elemanları

- APDU (Uygulama protokolü veri yapısı)
  - Okuyucu ile akıllı kart arasındaki veri iletiminin sağlandığı paket yapısıdır
  - APDU işlemleri için bir üst modeldir

- APDUOperation 
  - Modelleme ortamı üzerinde herhangi bir APDU işlemini tanımlayabilmek için tüm öğeleri içerir
  - commandAPDU ve responseAPDU elemanları bu eleman içinde tanımlanabilmektedir.
  - Gelen bir APDU paketinin nasıl işleneceği ve hangi işlemleri çalıştıracağı ile ilgili bilgileri tutabilmektedir.

 Including

Akıllı Kart Metamodel Elemanları

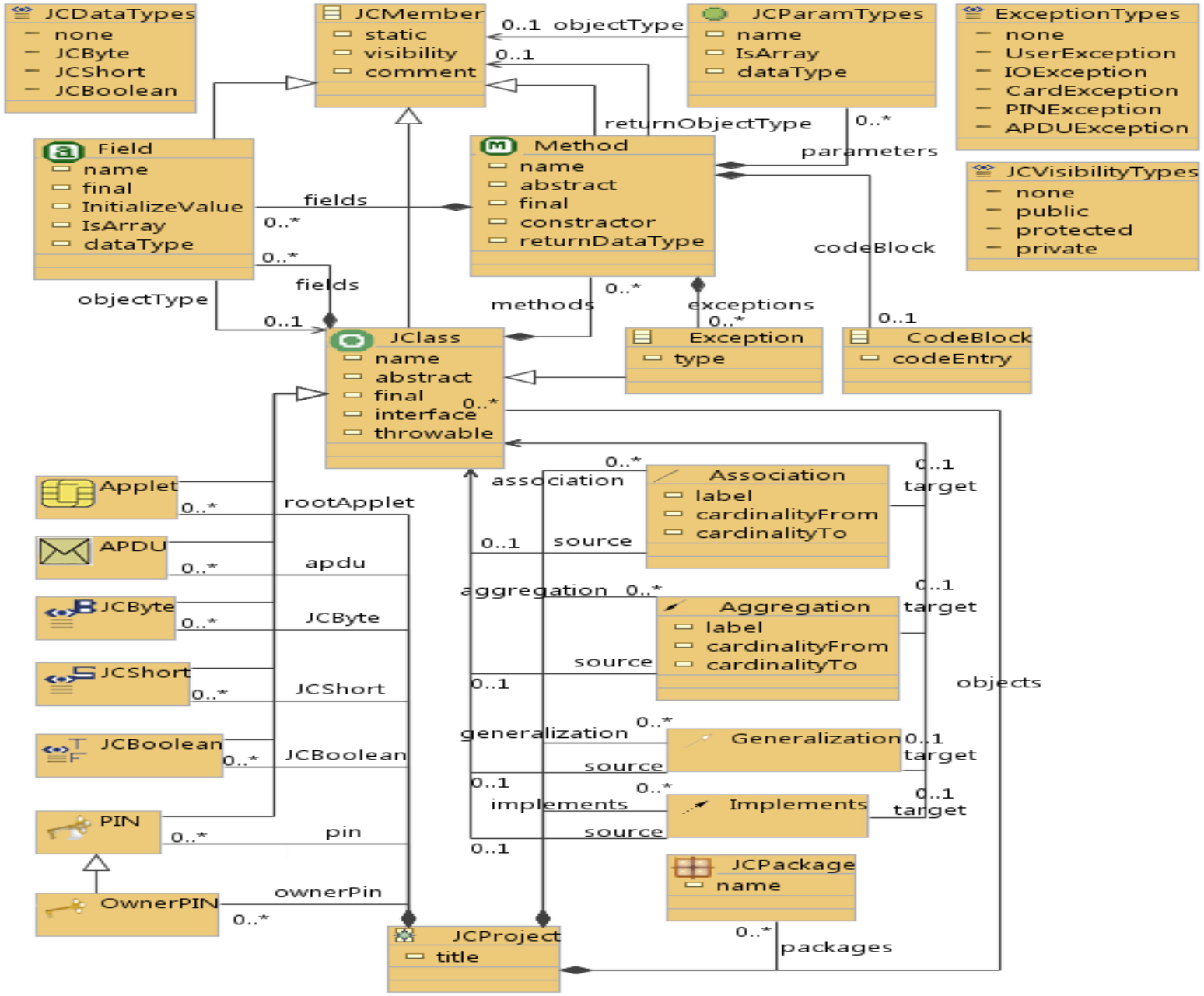
- PIN 
  - Akıllı Kart programlarına yetkili erişim sağlamak için şifre tanımlarının yapılabildiği bir ara yüz sağlar.
  - Okuyucu ile akıllı kart arasında bir bağlantı açılmadan önce *PIN* değerinin doğrulaması yapılmaktadır
  - Okuyucu *PIN* değerini içeren bir *APDU* paketini gönderir ve akıllı kart içindeki uygulama bu değerin kontrolünü yapar
  - Doğrulama ve pin değerini ilkleme komutları otomatik olarak üretilir
  - *Application* elemanı ile arasında *Access_key*  ilişkisi kurulmalıdır

Akıllı Kart Metamodel Elemanları

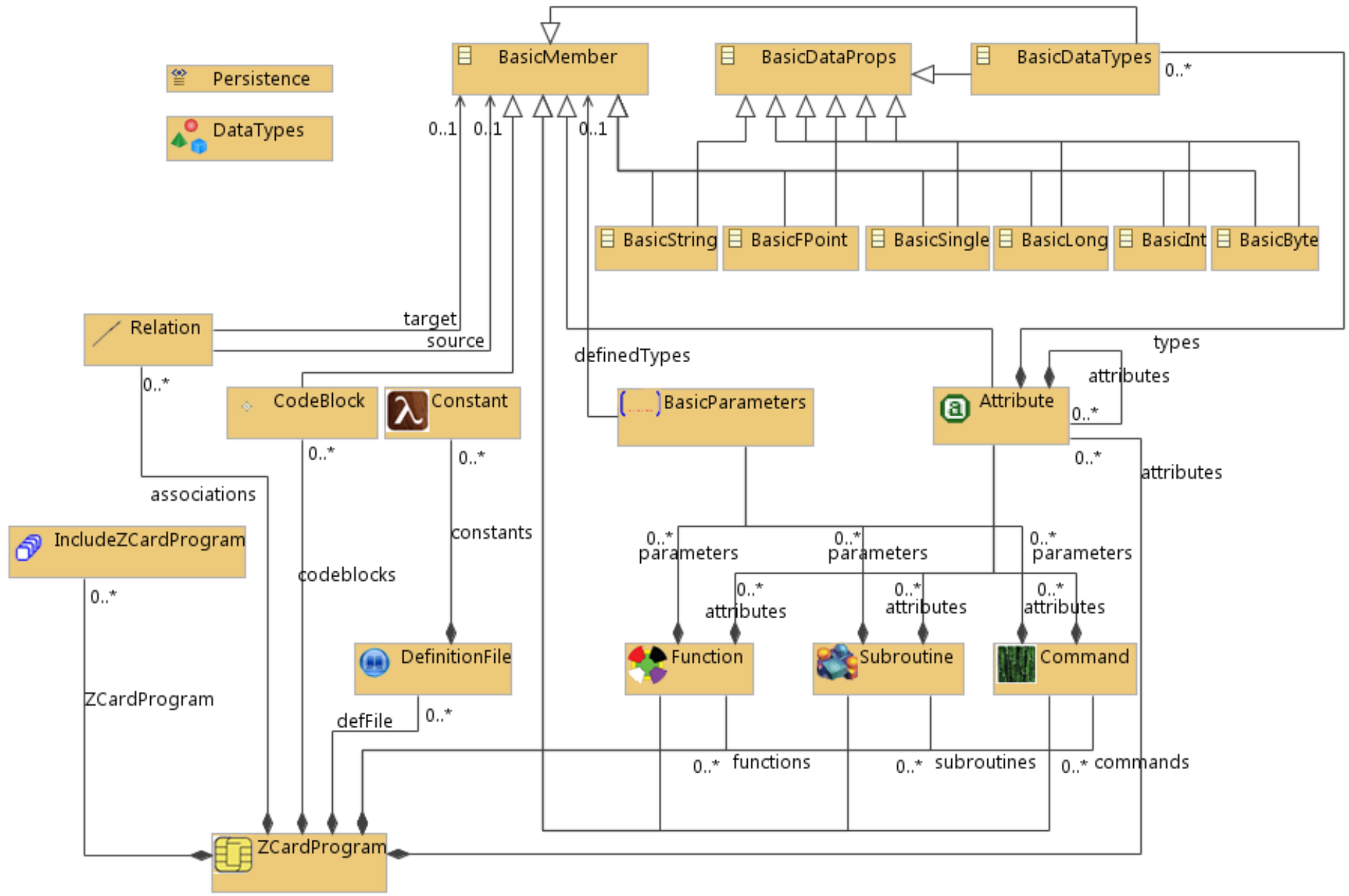
- *Constant*  ve *Data*  elemanları veri tipi tanımlamaları yapmak amacıyla kullanılmaktadır.
- *SCDataTypes*
  - *number*
  - *numberArray*
  - *string*
  - *boolean*
  - *byte*
- Kullanıcı tanımlı veri tipleri için *DataUnit* 
  - Farklı veri tiplerini bir araya toplayarak yeni bir veri tipi oluşturulabilmektedir.  Instance
- *Akıllı kart metamodelindeki tüm veri tipleri platforma özgü modellerde uygun veri tiplerine dönüştürülmektedir.*

Platforma Özgü Java Card ve Basic Card Modelleri

- Platforma Özgü Metamodel
  - Otomatik kod üretiminden önce, akıllı kart metamodelinden dönüşümlerin yapıldığı seviye
  - Dile bağlı özellikler içermesi
  - Platformlar hakkında bilgisi olan geliştiriciler için modellenen uygulamayı daha çok geliştirebilme imkanı



- Hazırlanan Java Card Metamodeli



- Hazırlanan Basic Card Metamodeli

Cüzdan Uygulaması

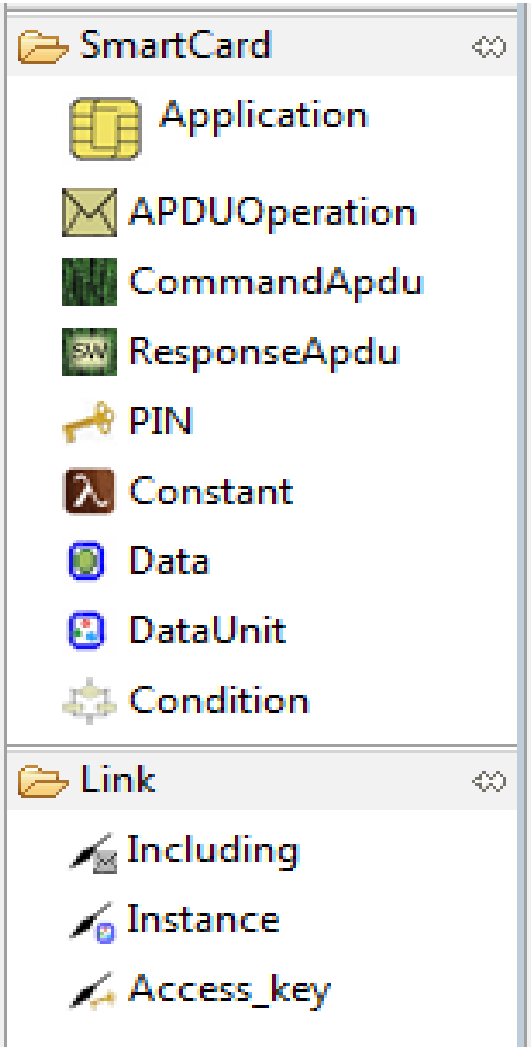
- Basit elektronik cüzdan komutları
  - Para düşme (Debit), Bakiye öğrenme (getBalance)
  - İndirim (Discount), Kredi yükleme (Credit)
- Her komut için bir *APDUOperation* işlemi
- Programda kullanılabilecek *Constant* tanımları
  - Çalıştırılan komutlar için geri dönüş hata değerleri
  - maxBalance, maxPinSize, pinTryLimit gibi tanımlar

Akıllı Kart Programlarının Modellenmesi

- Modelleme ortamı Eclipse platformu üzerinde Grafiksel Modelleme (GMF) araçları kullanılarak tanımlanmıştır.
- Geliştiriciler, akıllı kart sistemlerini modelleyebilmek amacıyla görsel editör ortamını kullanabilmektedirler.
- GMF üzerinde metamodel elemanları için görsel özellikler ve kısıtlar belirlenmiştir.

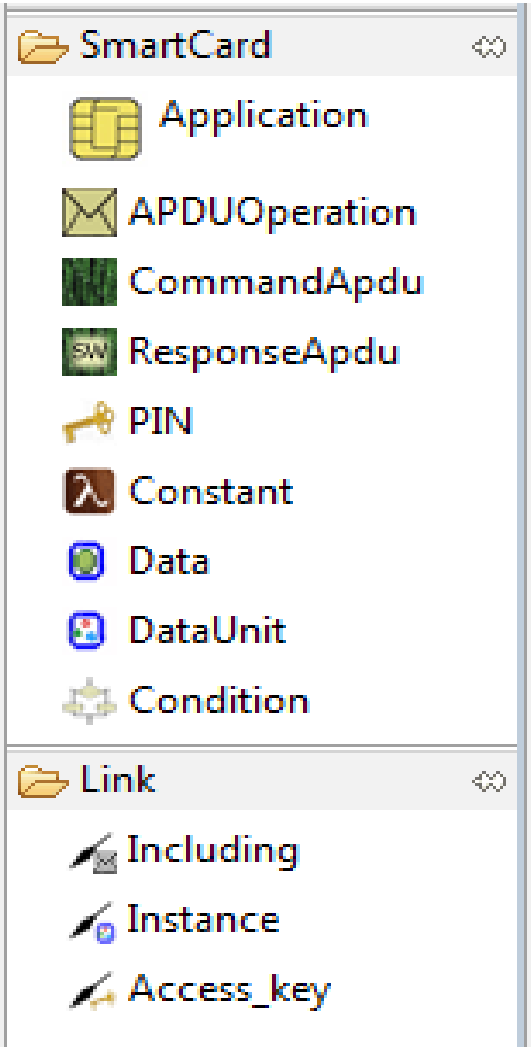
Akıllı Kart Programlarının Modellenmesi

- Akıllı Kart programı modelleyebilmek için gerekli olan elemanlar bir palet üzerinde tanımlanmıştır.
- Yanlış bir ilişki kurulmasını engellemek ve tasarlanan modellerin tutarlı olması sağlamak için bazı kısıtlar bulunmaktadır.
  - Command ve response APDU elemanları sadece APDUOperation elemanı içerisinde tanımlanabilmektedir.
  - İlişki okları sadece ilgili elemanlar arasında oluşturulabilmektedir.



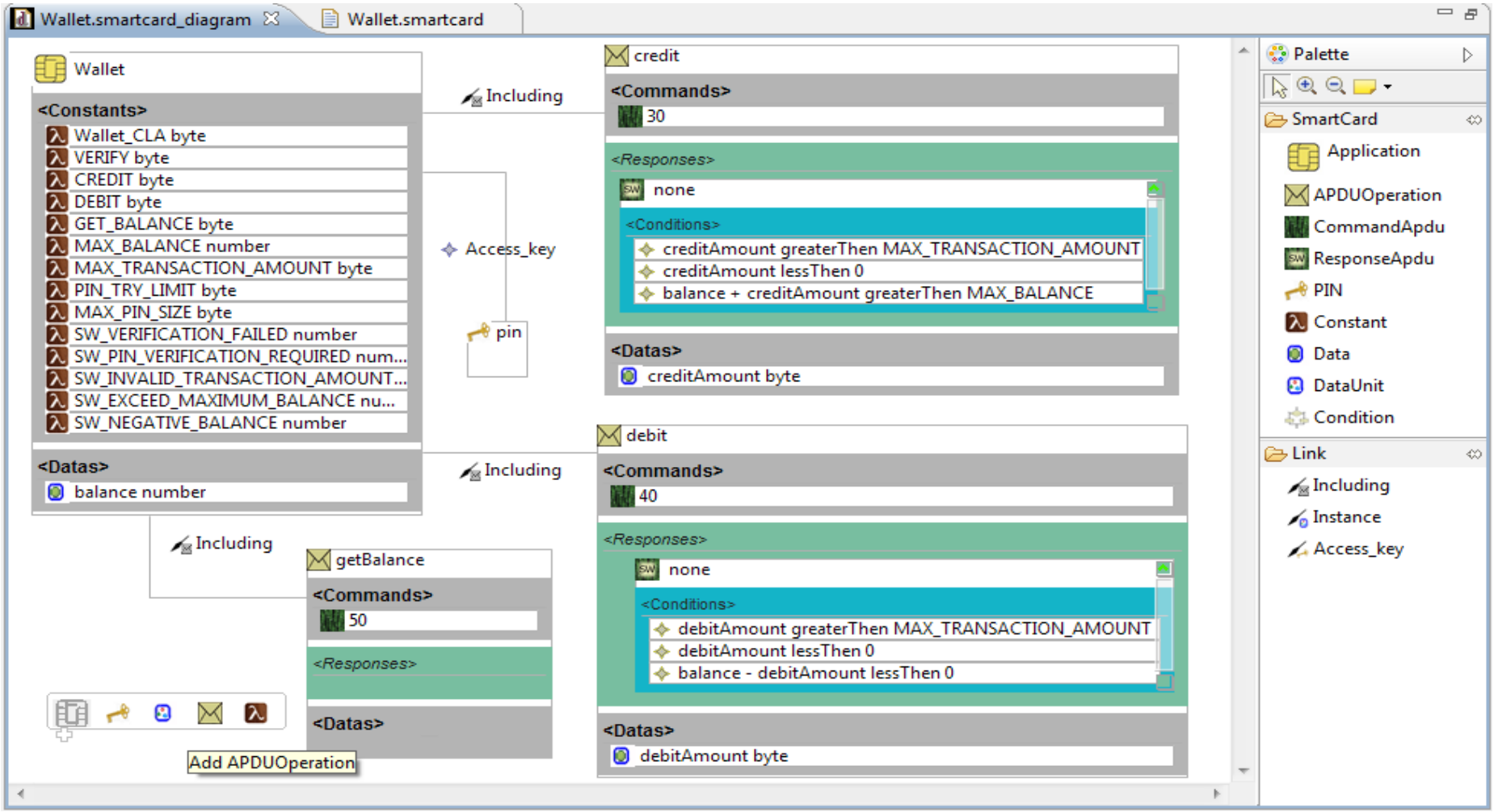
Akıllı Kart Programlarının Modellenmesi

- *Constant* elemanı *Application* içinde ya da *DataUnit* içinde tanımlanabilir.
- *Constant* ve *Data* elemanları ilgili olmadıkları bir alanda kullanılamazlar.
- Bir adet *Application* elemanı tanımlanabilir.



- Bir cüzdan uygulaması nasıl modellenir?

Akıllı Kart Programlarının Modellenmesi



❖Akıllı Kart modelleme araçları ile hazırlanan elektronik cüzdan modeli

Modeller arası Dönüşüm (M2M / PIM to PSM)

- Modeller arası dönüşüm kuralları ATL dili ile geliştirilmiştir.
- Akıllı kart örnek modelinden, platform bağımsız modellere(Java Card PSM, Basic Card PSM) ayrı ayrı dönüşüm kuralları yazılmıştır.
- Grafiksel modelleme arayüzü ile oluşturulan örnek modeller için otomatik olarak oluşturulan Ecore formatı, ATL dönüşüm kurallarına girdi olarak verilmektedir.

PIM to PSMs (M2M)

Platform-Bağımsız Modelden Platforma-Özgü Modellere

Akıllı Kart Metamodeli Platform Bağımsız Metamodeli	**Java Card Metamodeli**	**Basic Card Metamodeli**
SCProject	JCProject	ZCardProgram
Application	Applet	DefinitionFile
APDUOperation (command – response)	Method	Command
PIN	PIN	Attribute
Constant	Field	Constant

❖Akıllı Kart Modelinden Platforma Özgü Modellere Dönüşümler

PIM to PSMs (M2M)

SmartCard2JavaCard.atl

```
rule SmartCard2JavaCard{
    from
        smartCard : MM!SCProject
    to
        javaCard : MM1!JCProject(
            title <- smartCard.title,
            rootApplet <- Set{MM ! Application.allInstances()},
            ownerPin <-Set{MM!PIN.allInstances()},
            aggregation <- Set{MM!Access_key.allInstances()})}


rule Application2Applet{
    from
        appl : MM!Application
    to
        app_let : MM1 ! Applet(
            name <- appl.name,
            fields<- Sequence{appl.constants, appl.datas},
            methods<- appl.getAssociations())}
```

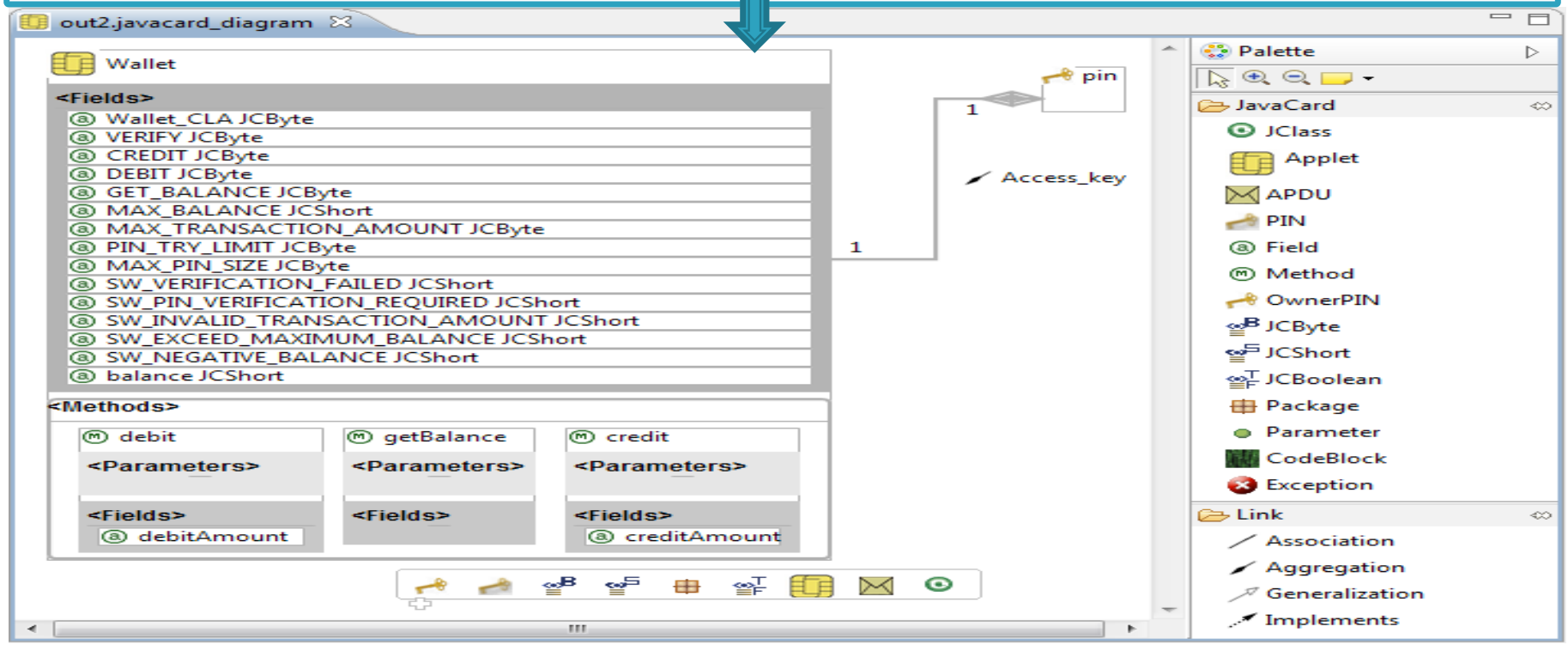
PIM to PSMs (M2M)

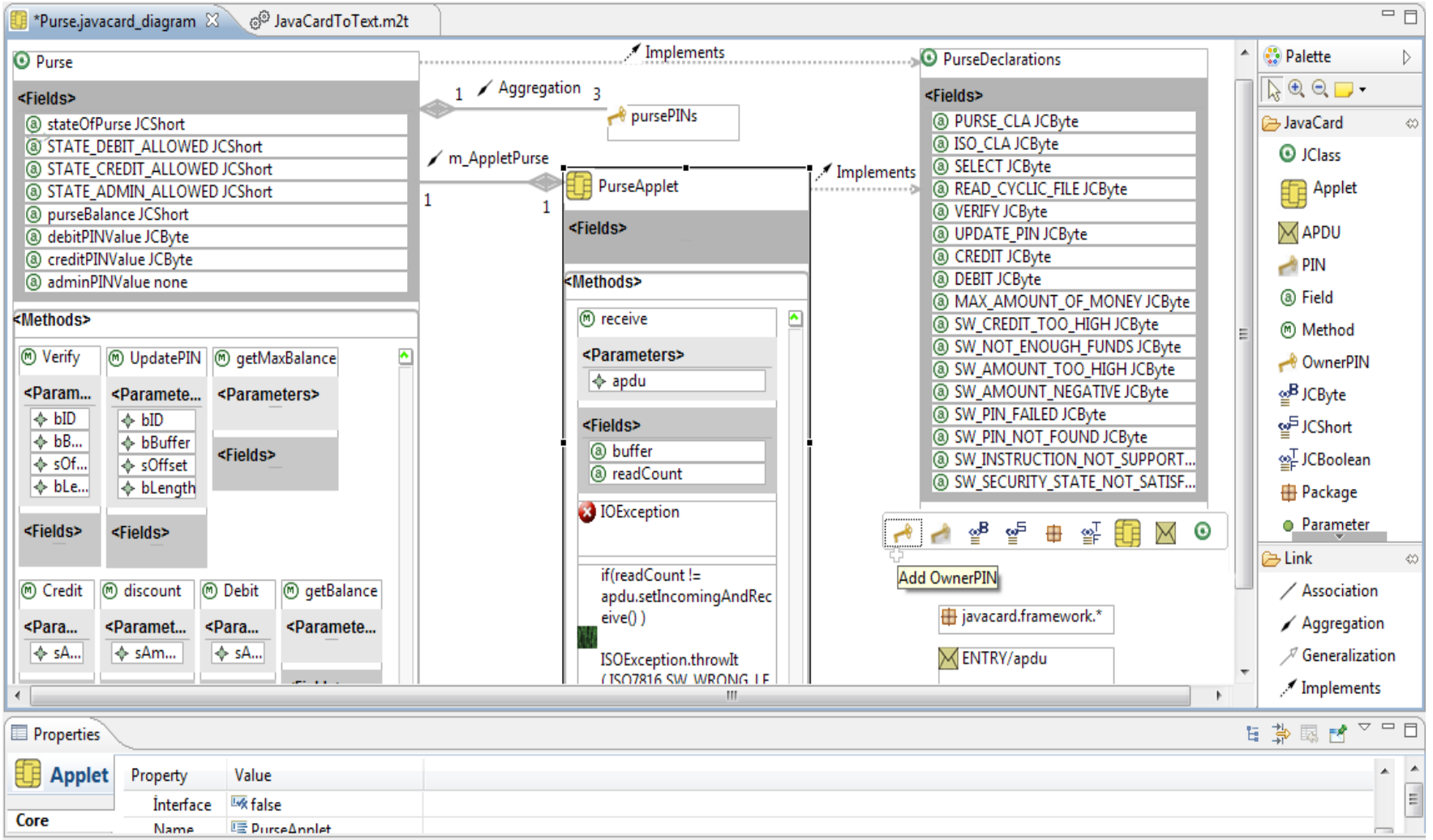
```
<?xml version="1.0" encoding="UTF-8"?>
<SCard:SCProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:SCard="org.smartcard.com" title="bank.purse">
  <apl name="Wallet" AID="">
    <datas name="balance" type="number"/>
    <constants name="Wallet_CLA" type="byte" value="0xB0" comment="code of CLA byte in the command
APDU header"/>
    <constants name="VERIFY" type="byte" value="0x20" comment="codes of INS byte in the command
APDU header"/>
    <constants name="CREDIT" type="byte" value="0x30"/>
    <constants name="DEBIT" type="byte" value="0x40"/> . . .
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<JCard:JCProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:JCard="org.javacard.com"
title="bank.purse">
  <rootApplet name="Wallet">
    <methods visibility="private" name="credit ">
      <fields static="true" comment="" name="creditAmount" final="true" InitializeValue="" dataType="JCByte"/>
    </methods>
    <methods visibility="private" name="debit">
      <fields static="true" comment="" name="debitAmount" final="true" InitializeValue="" dataType="JCByte"/>
    </methods>
    <methods visibility="private" name="getBalance"/>
    <fields static="true" comment="" name="Wallet_CLA" final="true" InitializeValue="0xB0"
dataType="JCByte"/>
    <fields static="true" comment="" name="VERIFY" final="true" InitializeValue="0x20" dataType="JCByte"/>
    <fields static="true" comment="" name="CREDIT" final="true" InitializeValue="0x30" dataType="JCByte"/>
    <fields static="true" comment="" name="DEBIT" final="true" InitializeValue="0x40" dataType="JCByte"/> . . .
```

PSM aşamasında Modelleme

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<JCard:JCProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:JCard="org.javacard.com"
title="bank.purse">
  <rootApplet name="Wallet">
    <methods visibility="private" name="credit ">
      <fields static="true" comment="" name="creditAmount" final="true" InitializeValue="" dataType="JCByte"/>
    </methods>
    <methods visibility="private" name="debit">
      <fields static="true" comment="" name="debitAmount" final="true" InitializeValue="" dataType="JCByte"/>
    </methods>
    <methods visibility="private" name="getBalance"/>
    <fields static="true" comment="" name="Wallet_CLA" final="true" InitializeValue="0xB0"
dataType="JCByte"/>. . .
```

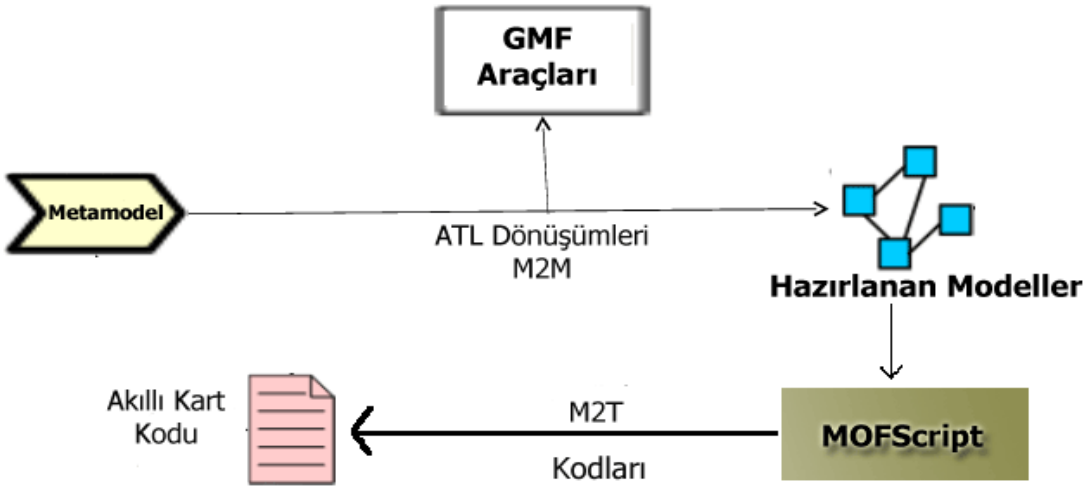




- Java Card modelleme araçları ile hazırlanan başka bir elektronik cüzdan modeli

Akıllı Kart Uygulamaları için Otomatik Kod Üretme

- Modellerden kod üretme amacıyla geliştirilen dönüşüm kodları MOFScript dili ile yazılmıştır.
  - MOFScript, Eclipse içinde bir araç olarak kullanılabilmekte ve geliştirilen modellerden herhangi bir anda kod üretilebilmektedir.
  - Hazırlanan model elamanlarının ve ilişkilerinin Ecore formatında tutulduğu dosya, MOFScript için bir girdi olarak algılanmaktadır.



MOFScript Dönüşüm Kuralları ve Üretilen Kodlar

Java Card MofScript Kodları

```
self.rootApplet->forEach(c:ec.JClass) {
          '
          public class ' c.name ' extends Applet {
            '
c.fields->forEach(c2:ec.Field) {
if (c2.visibility == "none") {
            '
            public '
} else {
                  '
                    '
c2.visibility ' '}
if (c2.static) {
     ' static '}
if (c2.final) {
     ' final '}
if (c2.IsArray) {
     var aggrRelation:List
     self.aggregation->forEach(c5:ec.Aggregation) {
     if ((c5.source.name = c.name)) {
        if (c5.target.name = c2.name) {
           aggrRelation.add(c5)}}}
     if (aggrRelation.size() > 0) {
           aggrRelation->forEach(c5:ec.Aggregation){
c5.target.name ' ' c2.name '[] = new ' c5.target.name'[' c5.cardinalityTo
'];'}}
```

```
package bank.purse;

  import javacard.framework.*;
  import javacardx.framework.*;

  public class Wallet extends Applet {
    public  static  final byte Wallet_CLA = 0xB0; // code of CLA byte in the command APDU header
    public  static  final byte VERIFY = 0x20; //
    public  static  final byte CREDIT = 0x30; //
    public  static  final byte DEBIT = 0x40; //
    public  static  final byte GET_BALANCE = 0x50; //
    public  static  final short MAX_BALANCE = 0x7FFF; // maximum balance
    public  static  final byte MAX_TRANSACTION_AMOUNT = 127; //    maximum transaction amount
    public  static  final byte PIN_TRY_LIMIT = 0x03; // maximum number of incorrect tries
    public  static  final byte MAX_PIN_SIZE = 0x08; //
    public  static  final short SW_VERIFICATION_FAILED = 0x6300; //
    public  static  final short SW_PIN_VERIFICATION_REQUIRED = 0x6301; //
    public  static  final short SW_INVALID_TRANSACTION_AMOUNT = 0x6A83; //
    public  static  final short SW_EXCEED_MAXIMUM_BALANCE = 0x6A84; //
    public  static  final short SW_NEGATIVE_BALANCE = 0x6A85; //
    OwnerPIN pin;
    short balance;

    public static void install(byte[] bArray, short bOffset, byte bLength) {
     new Wallet();
    }

    public Wallet() {
        pin = new OwnerPIN((byte)(5),(byte)(8));
      pin.update(/*write your pin value variable here*/, (short)(0), (byte)(/*write your size of pin here in bytes*/));
      register();
    }
    public void process(APDU apdu) throws ISOException{

        // APDU object carries a byte array  ( buffer )  to
        // transfer incoming and outgoing APDU header
        // and data bytes between card and CAD
        // At this point, only the first header bytes
        //  [ CLA, INS, P1, P2, P3 ]  are available in
        // the APDU buffer.
        // The interface javacard.framework.ISO7816
        // declares constants to denote the offset of
        // these bytes in the APDU buffer

         byte[] buffer = apdu.getBuffer();
         //Examination of the buffer.
         switch(buffer[ISO7816.OFFSET_INS])
         {

 case GET_BALANCE: getBalance ( apdu ) ;
 return;
 case DEBIT: debit ( apdu ) ;
 return;
 case CREDIT: credit ( apdu ) ;
 return;
 case VERIFY: verify ( apdu ) ;
 return;
 default: ISOException.throwIt
  ( ISO7816.SW_INS_NOT_SUPPORTED ) ;
        }

    }
```

Üretilen Kodun İçeriği

- PIM seviyesinde modelleme ile Java Card koduna dönüşümler
  - Process, Install gibi temel Java Card metotları
    - Küçük farklılıklar haricinde, Java Card kodunda bulunması gereken temel işlemler ile çalışan seviyede kod üretimi
  - Kullanıcı tanımlı metotlar
    - Metot imzalarının tümü
    - Kullanıcı tarafından girilen koşulların dile özgü kullanımları
    - Standart bir metot içinde bulunabilecek küçük kod blokları
  - Sabit ve Değişken veri tipleri
    - Tamamı
  - Pin İşlemleri
    - Temel işlemleri yerine getirecek metot ve parametrelerin üretimi

Üretilen Kodun İçeriği

- PIM seviyesinde modelleme ile Basic Card koduna dönüşümler
  - Paket alış verişini sağlayacak "Command" prosedürleri
    - Prosedür imzaları, aldığı parametreler, gövde kısmındaki koşul içeren kodlar ve geri dönüş değerleri
  - Kullanıcı tanımlı metotlar (Subroutine ve Function)
    - Sadece PSM seviyesinde
  - Sabit ve Değişken veri tipleri ile sabitlerin tanımlandığı diğer proje dosyaları
    - Tamamı

Plan ve Sonuç

- Geliştirmesi devam eden çalışmamızın içeriği:
  - Üretilen kodun verimliliği ve doğruluk testleri
  - Otomatik üretilen kod miktarının artırılmasına yönelik çalışmalar
    - Bu sayede otomatik üretilen kod üzerinde en az seviyede değişiklik yapılması
- İleriye Dönük Düşünceler
  - Farklı akıllı kart tiplerine destek
    - Platform Bağımsız Model bu genişlemeye olanak verecek şekilde tasarlandı

Teşekkürler

- Sorularınız?