ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Automata-Based Programming Technology Extension for Generation of JML Annotated Java Card Code

Andrey Klebanov, CTD, SPb SU ITMO

supervised by Anatoly Shalyto, Ph. D,
prof. CTD, SPb SU ITMO

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- Automata-based programming technology
- Java modelling language (JML)
- Approach description
- Case study
- Open questions

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- **Smart Cards**
- Automata-based programming technology
- Java modelling language (JML)
- Approach description
- Case study
- Open questions

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Smart Cards

- «Stupid» cards – cards with just magnetic stripe;
- Smarts cards – chip and memory are embedded:
  - Mobile and secure credit card size computers;
  - Very limited recourses – 1-4Kb RAM, 48-64Kb NVM (ROM) + 8-32Kb EEPROM;
  - Main domains of use are secure storage of data, business transactions, authentication, ...
  - Vendor specific, difficult to develop applications.

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Java Card

- Java platform for smart cards;
- Provides all the benefits of Java and also
  - Allows to abstract away from low-level features of different cards;
  - Applet isolation mechanism;
  - Post-issuance applet downloading, ...
- Java Card API 2.2.2 is a superset of Java API subset;
- Java Card 3.0 will be discussed in «Open questions» section.

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Formal methods for JC

Several reasons to attract formal methods researchers:

- Java Card domain of use, industry support;
- Complexity of updating;
- Relatively small, but real-world applications.

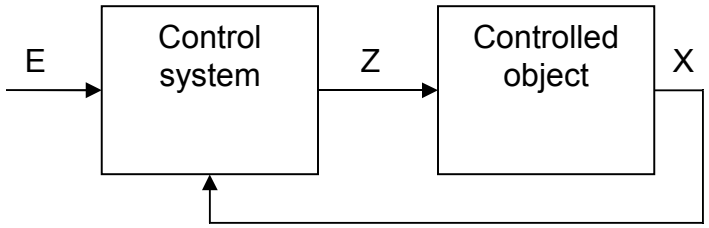
ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- **Automata-based programming technology**
- Java modelling language (JML)
- Approach description
- Case study
- Open questions

Automata-based programming overview

- Introduced by A. Shalyto in 1991;
- Sort of synchronous programming;
- Programs are treated as systems of automated controlled objects;
- Each system consists of control system and controlled objects;
- Control system - system of co-operating automata.



- X_i – input action;
- Z_i – output action;
- E – event;

ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

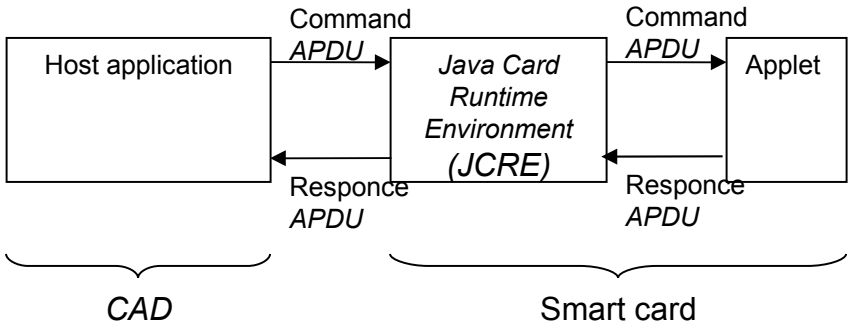
Automata-based programming benefits

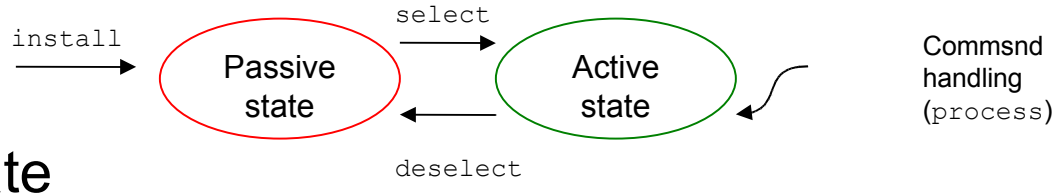
- Formally describes application logic and behaviour;
- Perfect solution for reliable application development for reactive and embedded systems;
- Defines two types of diagrams for application description – connectivity schema and transition graphs;
- Fully supported by the UniMod tool
  - Closes the gap between model and implementation via Java code generation;
  - Finite state machine validation.

Automata-based programming for Java Card

- Half-duplex communication channel, master-slave model;
- Event driven interaction
  - Host application – event provider;
  - Smart card – controlled object.
- Standard structure of applet, logic is incapsulated in one method.
- «Java Card applet is a state machine.»

Wikipedia





ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- Automata-based programming technology
- **Java modelling language (JML)**
- Approach description
- Case study
- Open questions

JML

- JML is a behavioural interface specification language;
- JML is based on design by contract, but extends it greatly;
- Designed to be used by Java programmers;
- Tailored to Java;
- Doesn’t require programs to be OO;
- A lot of tools are developed to support JML.

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

JML (cont.)

- Preconditions (`requires`), postconditions (`ensures`) and invariants (`invariant`);
- `\old(var)` – variable `var` value before method execution;
- Logical constructions (ex. implication) and `constaraint` consruction – constraints variable's value change in time;
- `pure` and `assignable` keywords.

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

JML (cont.)

- `private` fields could be declared as `spec_public`;
- Quantifiers – `\forall`, `\exists`;
- `\min`, `\sum` expressions;
- Allows to describe behaviour in exceptional situations;
- And much more!

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- Automata-based programming technology
- Java modelling language (JML)
- **Approach description**
- Case study
- Open questions

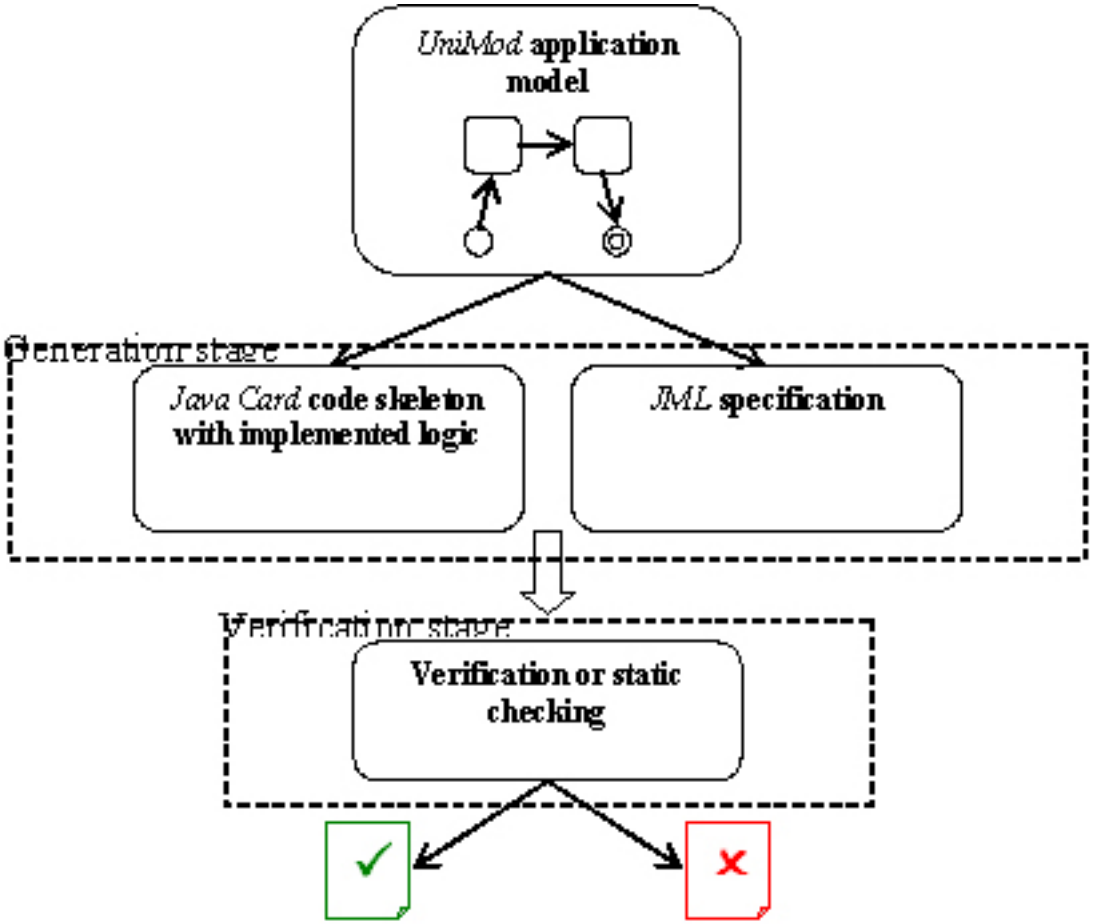
ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Approach description

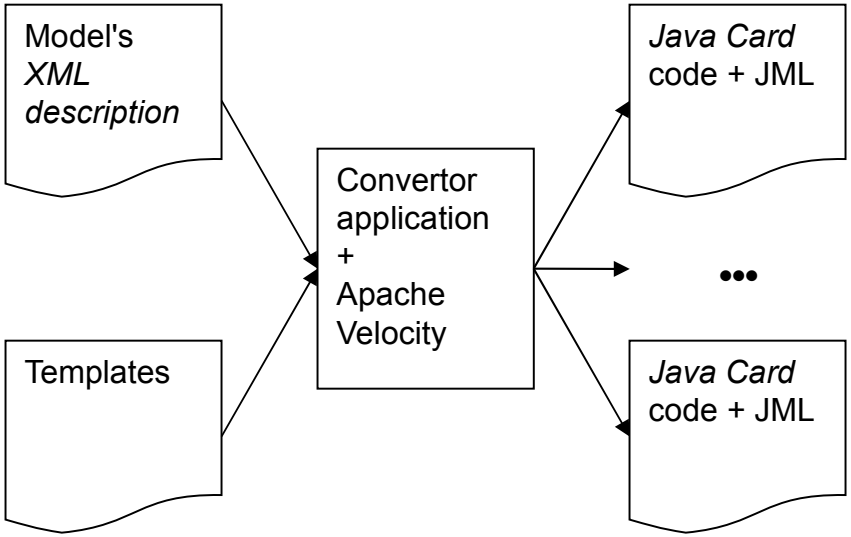
- Problem: Java Card code should be trustworthy and bug-free.
- Solution: automata-based programming + JML!
- Sub-problems to be solved:
  - Extend automata-based programming code generation technologies;
  - Convert state machine model to JML annotations;
  - Explore different verification tools designed to work with JML.

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Approach overview

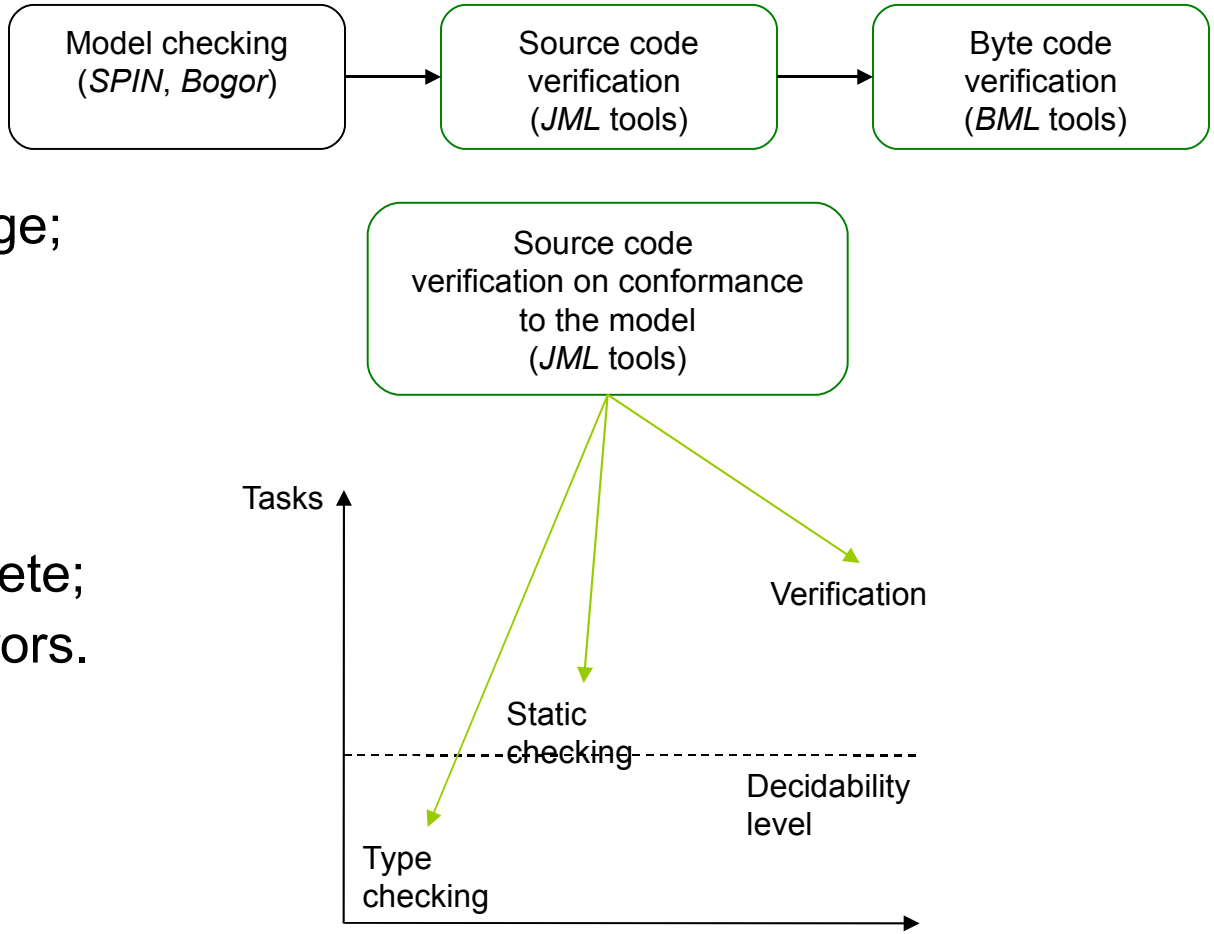


Annotated code generation stage



Verification stage

- *jmlc*
  - Fully automatic;
  - Full language coverage;
  - Doesn't prove errors absence.
- ESC/Java2
  - Fully automatic;
  - Not sound, not complete;
  - Good for common errors.
- KeY, Loop, Jack, ...
  - Powerful;
  - Interactive.

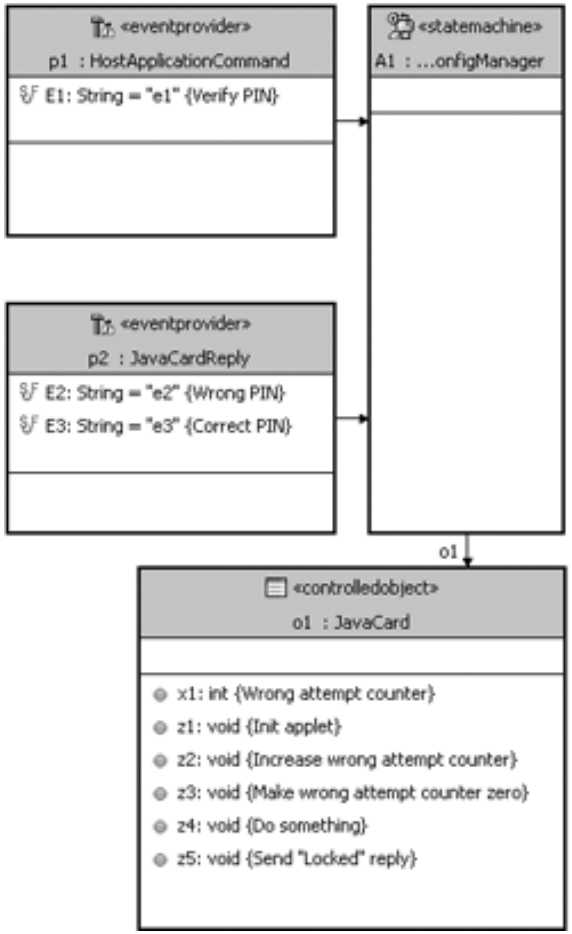


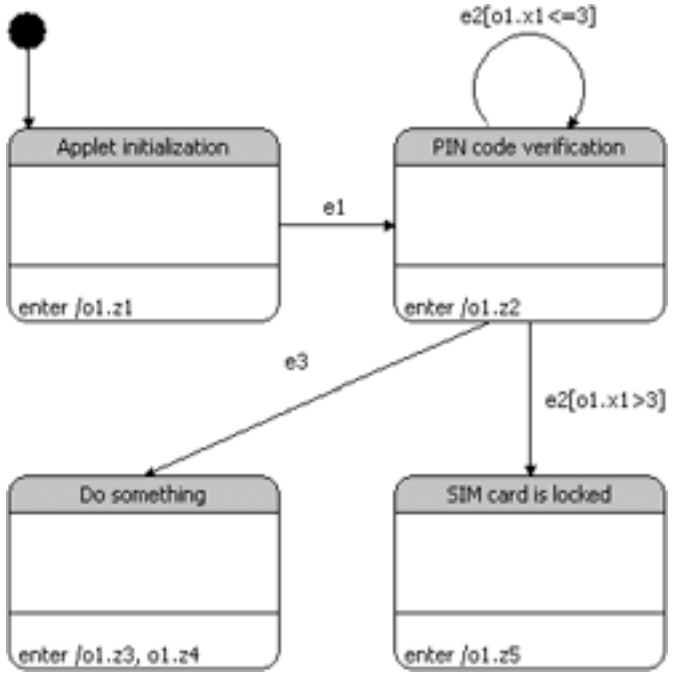
ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- Automata-based programming technology
- Java modelling language (JML)
- Approach description
- **Case study**
- Open questions

Case study – description





Case study – several results

- Convinient notation for commands vs. byte arrays;
-
```
/*@ invariant
 (state == APPLET_INITIALIZATION) ||
 (state == VERIFY_PIN) ||
 (state == DO_SOMETHING)          ||
 (state == SIM_CARD_IS_LOCKED);
@*/
```
- Precondition for the on enter to state *SIM card is locked* – `//@ requires x1 > 3;` (if `x1` has no side effects).

Case study – several results (cont.)

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Transitions between states:

```
/*@ constraint
     ((state == APPLET_INITIALIZATION) ==>
     (\old(state) == APPLET_INITIALIZATION)) &&
     ((state == VERIFY_PIN) ==> ((\old(state) == VERIFY_PIN) ||
     (\old(state) == APPLET_INITIALIZATION))) &&
     ((state == DO_SOMETHING) ==>
     ((\old(state) == VERIFY_PIN) ||
     (\old(state) == DO_SOMETHING))) &&
     ((state == SIM_CARDS_IS_LOCKED) ==>
     ((\old(state) == VERIFY_PIN) ||
     (\old(state) == SIM_CARDS_IS_LOCKED))) &&
     ((\old(state) == APPLET_INITIALIZATION) ==>
     ((state == VERIFY_PIN) ||
     (state == APPLET_INITIALIZATION))) &&
     ((\old(state) == VERIFY_PIN) ==>
     ((state == VERIFY_PIN) ||
     (state == DO_SOMETHING) ||
     (state == SIM_CARDS_IS_LOCKED))) &&
     ((\old(state) == DO_SOMETHING) ==>
     (state == DO_SOMETHING)) &&
     ((\old(state) == SIM_CARDS_IS_LOCKED) ==>
     (state == SIM_CARDS_IS_LOCKED));
@*/
```

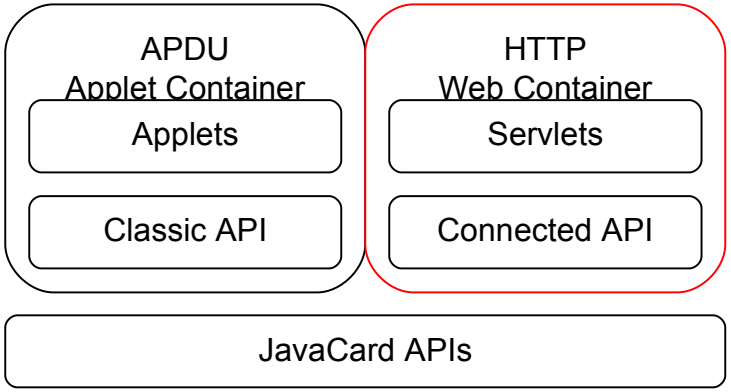
ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Outline

- Smart Cards
- Automata-based programming technology
- Java modelling language (JML)
- Approach description
- Case study
- **Open questions**

Open questions

- Java Card 3.0
  - Great new opportunities close to «big» Java!..
  - But possible problems for formal methods.



- Java ME
  - Midlets are running on constraint devices...
  - But much more powerful then smart cards.

Thank you!

Thank you!