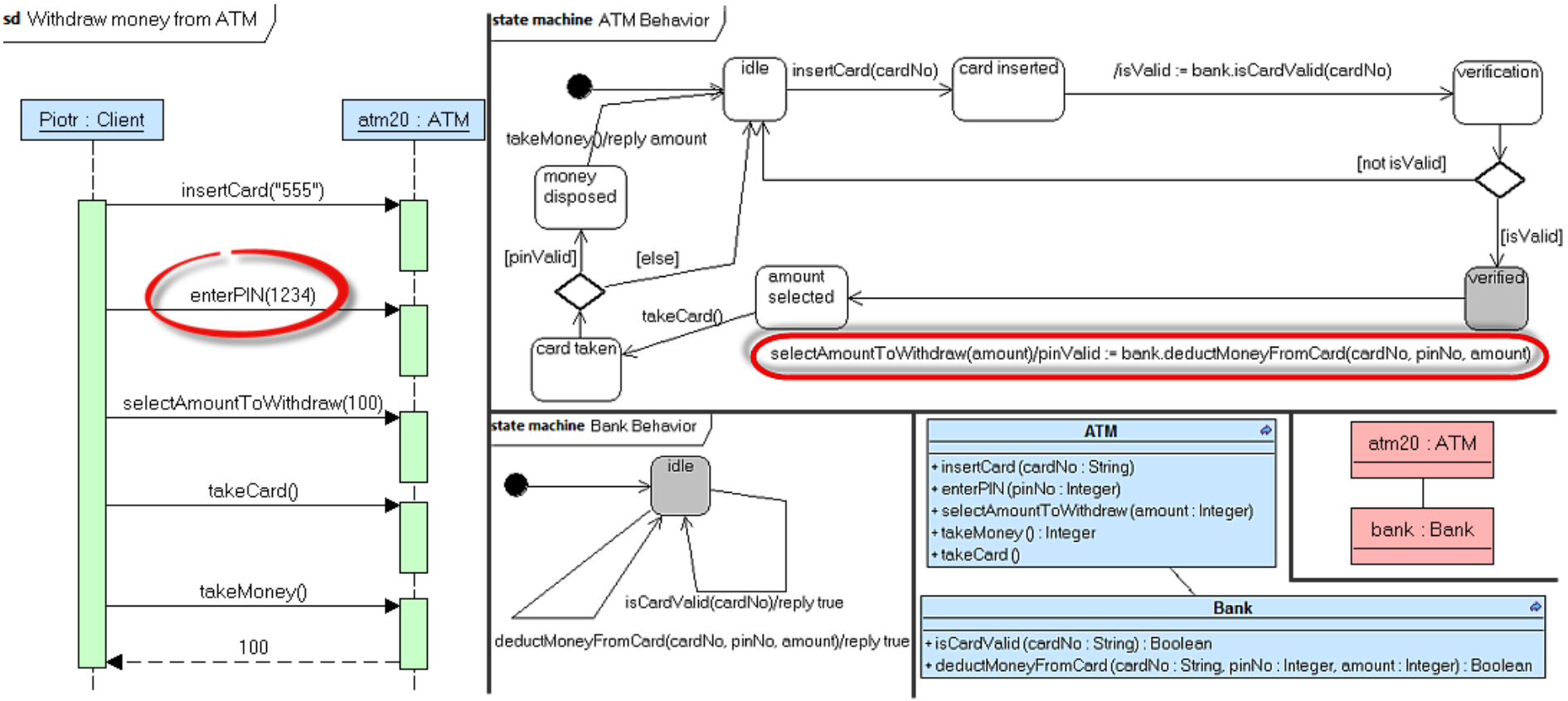
Checking consistency between interaction diagrams and state machines in UML models

Piotr J. Puczynski

DTU Informatics

June 12, 2012

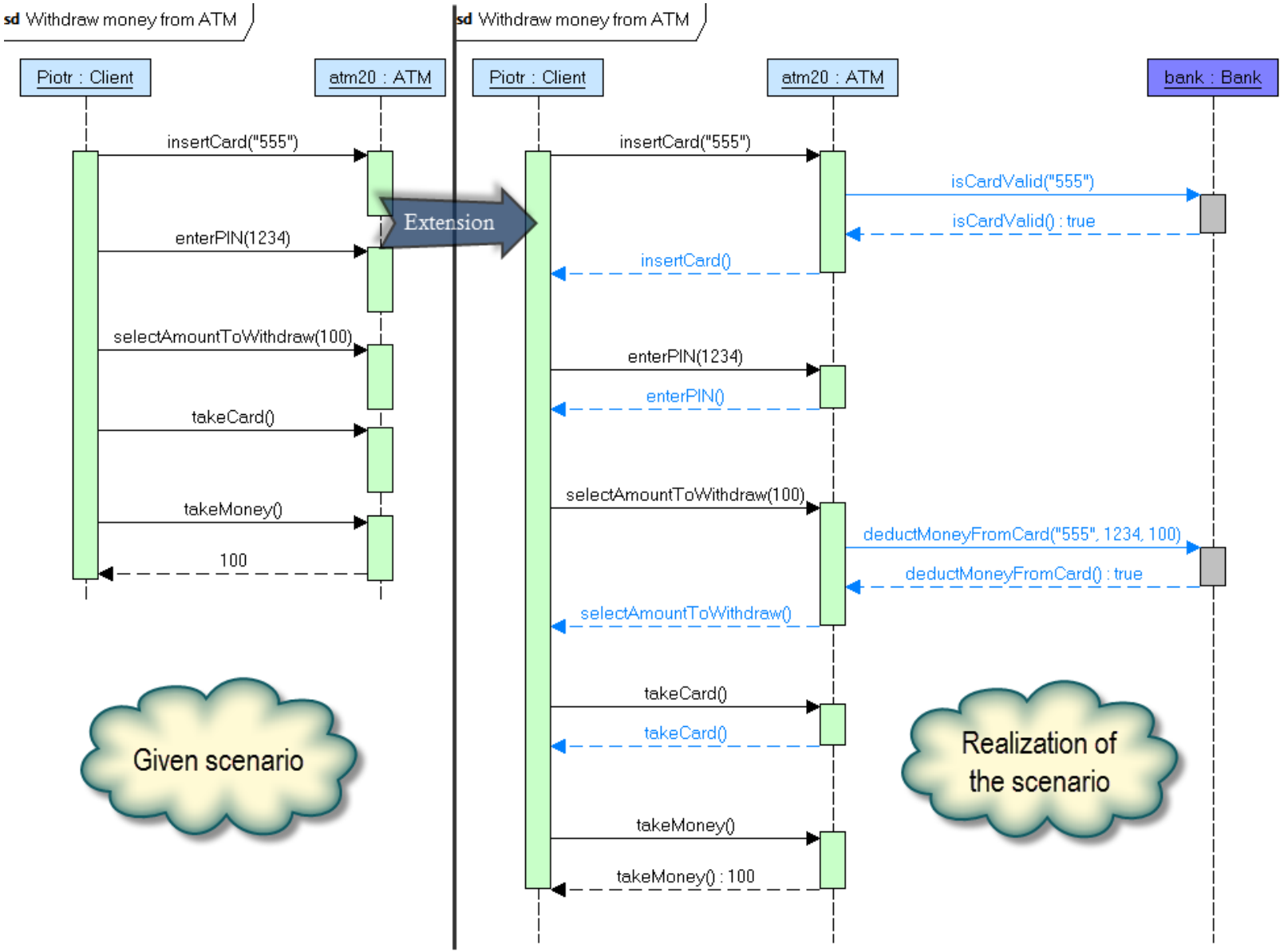
Inconsistency example (faulty ATM System)



Aim of the Master's project

- Check for consistency between (behavioral and protocol) state machines and interactions (sequence diagrams)
- Extending use case scenarios (sequence diagrams)
- Hints to a user during design

Extensions of scenarios



Related work

Two main groups of consistency checking techniques:

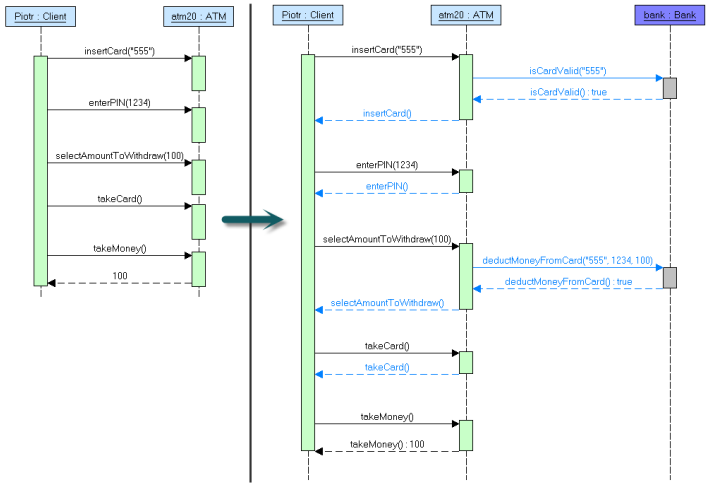
1. usage of intermediate representations
   - model checking
2. direct usage of UML models
   - simulation

- UML 1.x and no CASE tools support (98.4%)
- No extensions of sequence diagrams

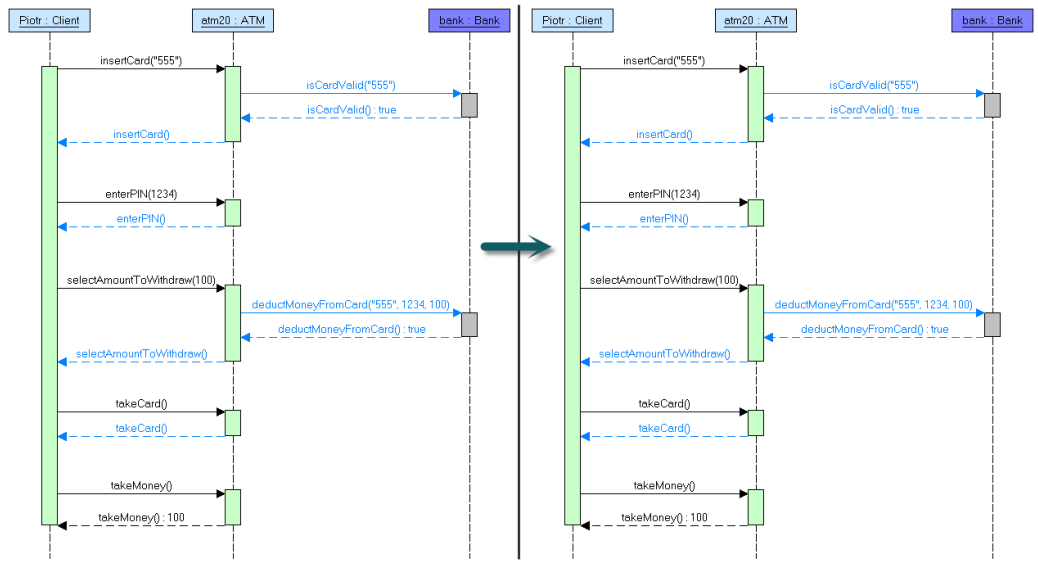
Checking consistency

1. Structural properties of model (incl. components, interfaces)
2. Sequence diagrams and behavioral state machines

a) Extension of given scenarios to realizations of the scenarios

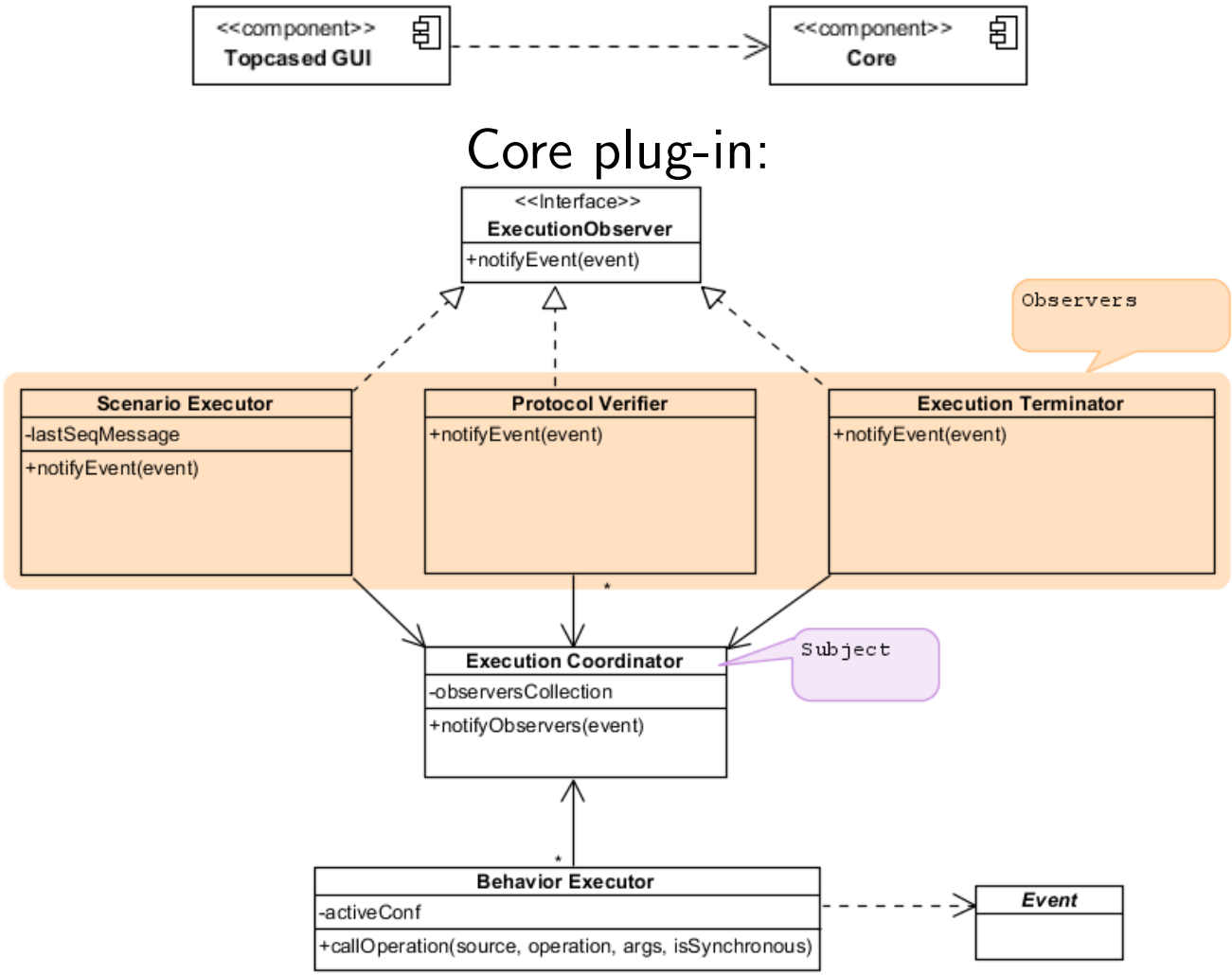


b) Validation of (already extended) realizations of scenarios

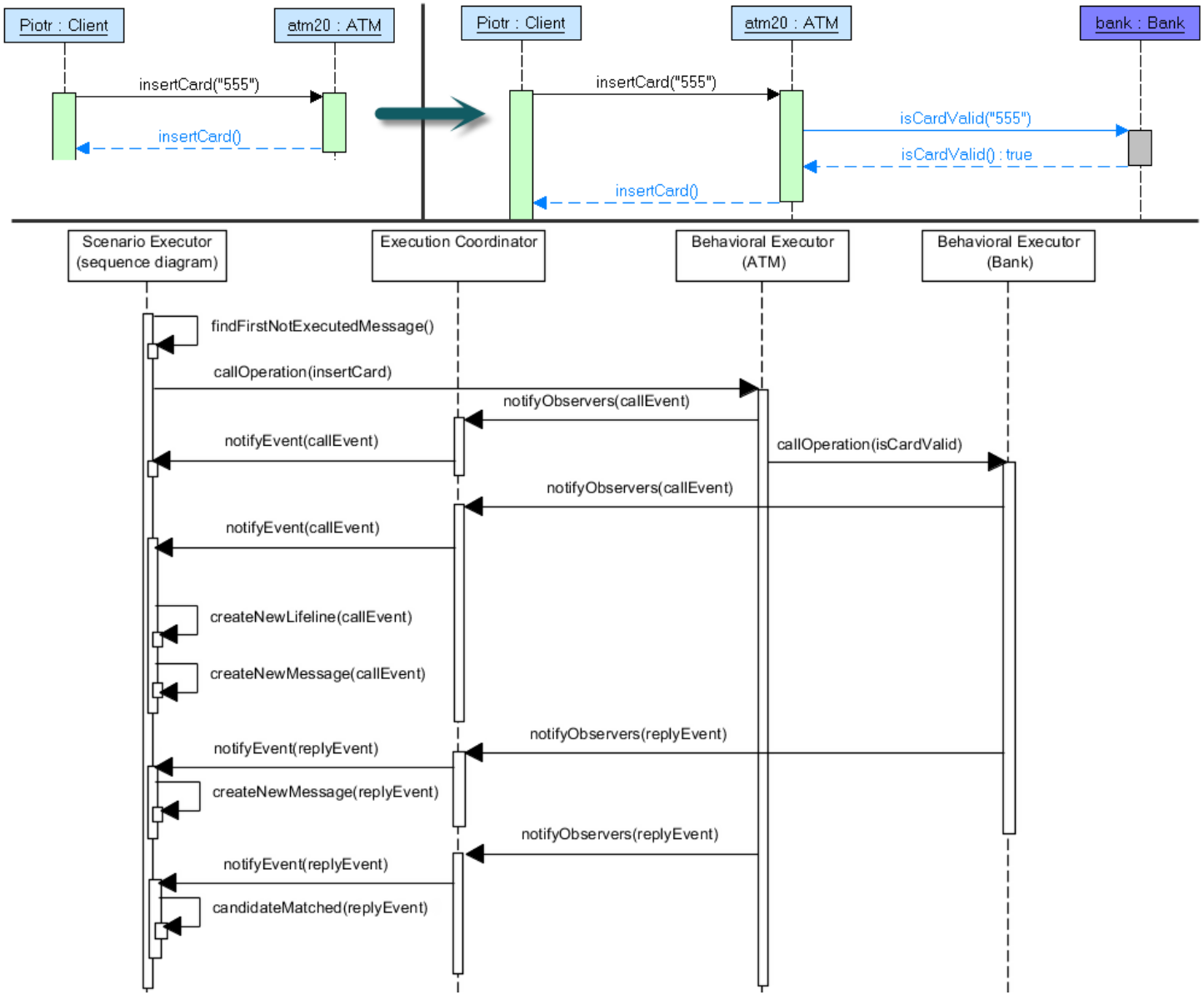


3. Protocol state machines and behavioral state machines

Design of the tool



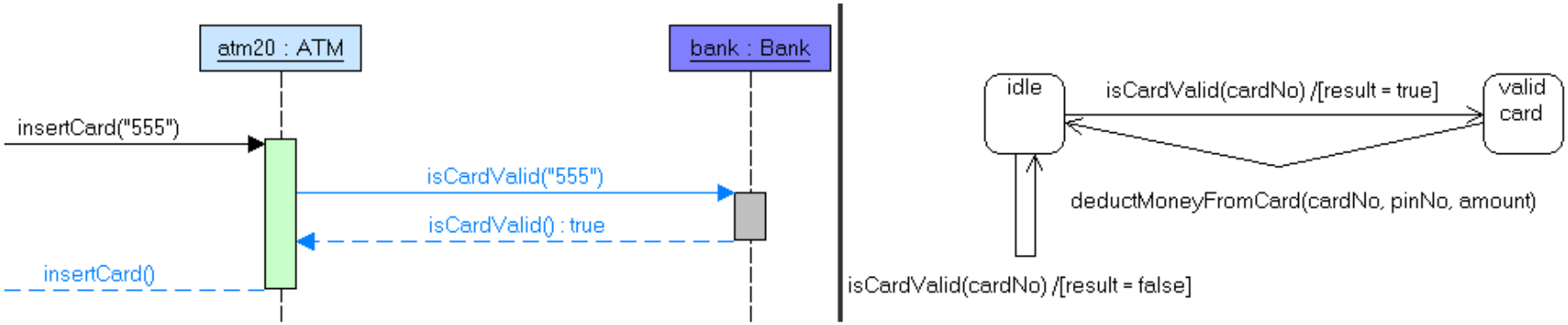
Extending and validating sequence diagrams



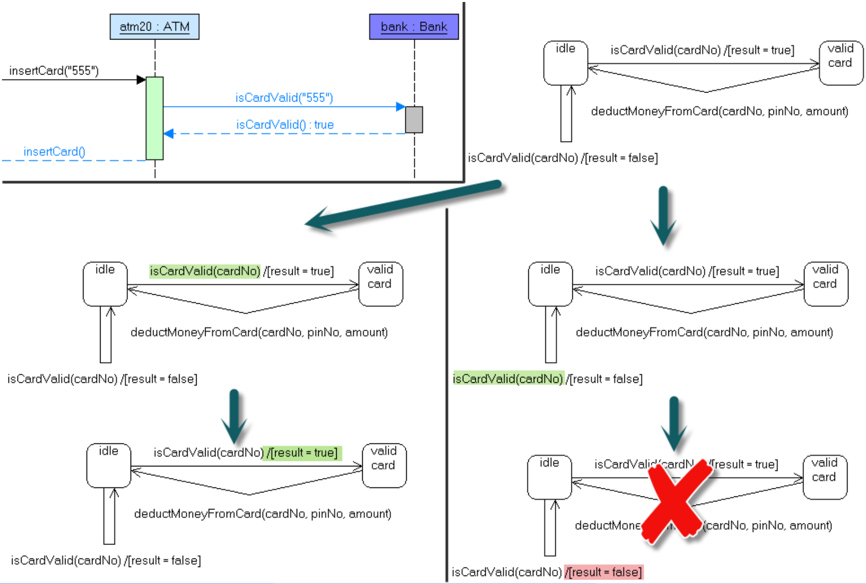
Behavioral state machines execution

```
1 callOperation() =
2     notifyObservers(callEvent)
3     execute(effects(enabledTransition(operation)))
4     execute(effects(completionTransitions()))
5     if isSynchronous
6         notifyObservers(replyEvent)
```

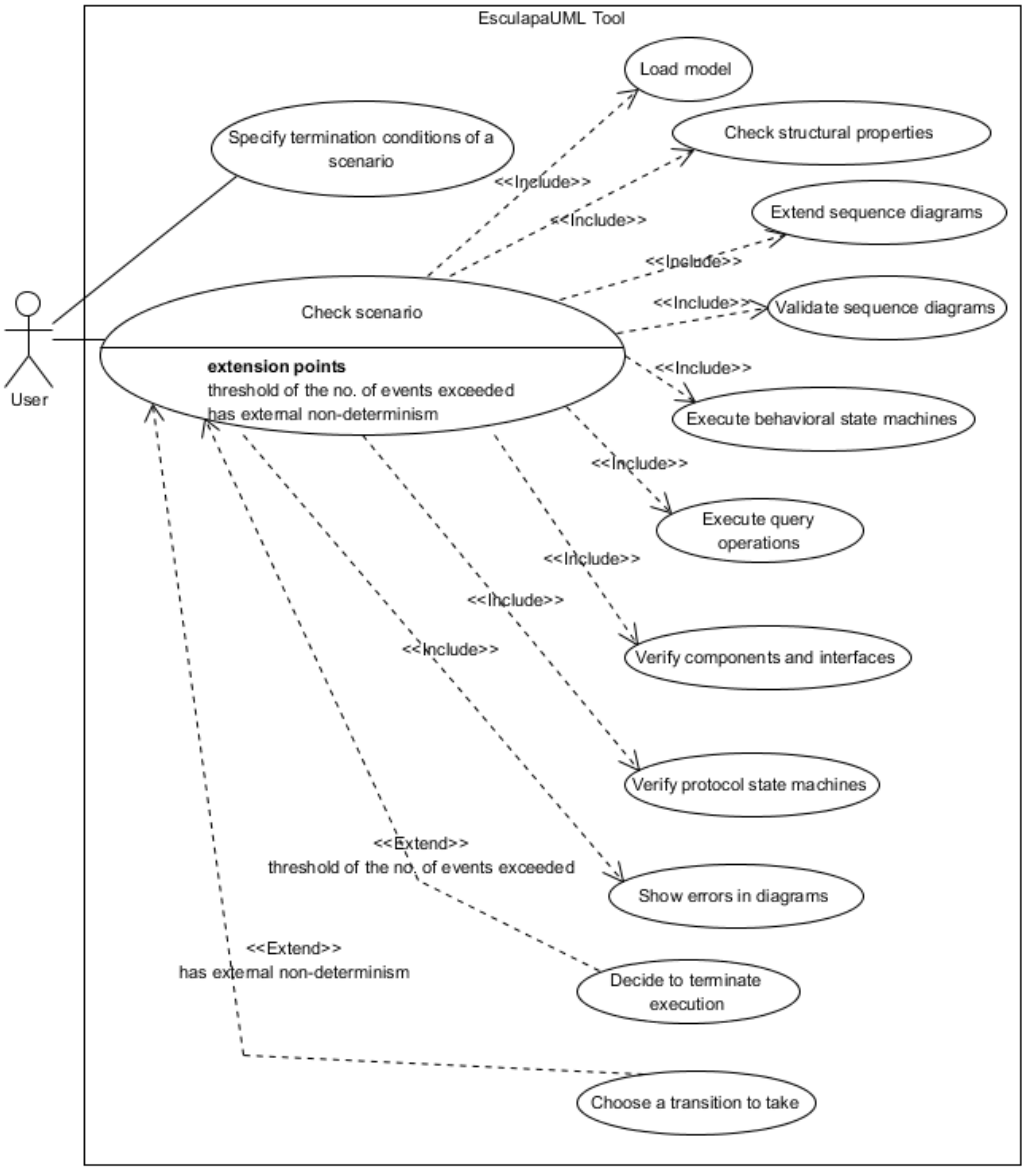
Verification of protocol state machines



Verification of protocol state machines



Example run in ATM system



Checks and error messages

- Support for **130** distinct error messages
- Examples:
  1. Instance `instanceName` is not ready to respond to an event `eventName`.
  2. Multiplicity check failed when trying to assign return value for operation `operationName` to value: `valueToAssign`.
  3. Provided interface `interfaceName` is not realized by any class in component `componentName`.

Conclusions and future work

- Checking consistency by realization of scenarios
- Simple Action Language (SAL)
- Statistics: 130 distinct error messages; 110 scenarios and test models; two models of the toll system

Conclusions and future work

- Checking consistency by realization of scenarios
- Simple Action Language (SAL)
- Statistics: 130 distinct error messages; 110 scenarios and test models; two models of the toll system
- Problems:
  - Semantics of run-to-completion
  - Termination of scenarios
  - 3rd party bugs

Conclusions and future work

- Checking consistency by realization of scenarios
- Simple Action Language (SAL)
- Statistics: 130 distinct error messages; 110 scenarios and test models; two models of the toll system
- Problems:
  - Semantics of run-to-completion
  - Termination of scenarios
  - 3rd party bugs

Future work:

- Support for more UML elements
- Creation and destruction in SAL
- More functionality (e.g. step-by-step simulation)
- Experimental evaluation