

SESAME: a model-driven approach for flexible test selection of industrial applications



SESAME
DEVELOPMENT METHODOLOGY
FOR EMBEDDED SYSTEMS

Benoît Ries
<http://lassy.uni.lu/users/BR>

LASSY Seminar – April 30, 2008

Outline

- Introduction
 - Test Selection Problem
 - PhD Objectives
 - Process Overview
- Current Result: SESAME Process
- SESAME Finalization Tasks



Issues

- Test Selection Problem
 - Exhaustive testing is impossible (in general):
 - Possibly very large number of traces, e.g. def. domains.
 - Possibly very long traces, e.g. loops.
 - Thus, test selection is a necessary activity
 - Amount and/or length of test cases must be reduced until reaching a reasonable limit
 - But how to specify the test selection?



Issues (cntd)

- State of practice of test selection specification
 1. Most often, test specifications enumerates the test cases (scenario-driven test selection)
 - + focus on “important” behaviors to be tested
 - - poor coverage of the overall system behavior
 2. Otherwise, automated test generation tools are used
 - + large coverage of the system behaviors
 - - test selection done informally after generation
 - - test selection specification is hidden in the generation process, test engineers don't know what has been tested



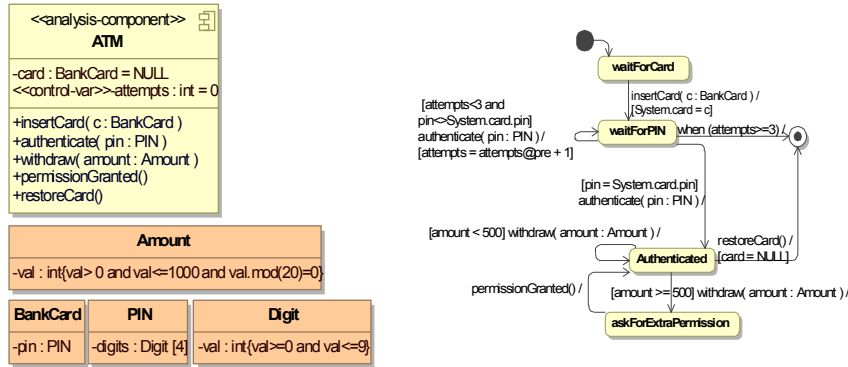
PhD Objectives

- Define the SESAME approach, a model-driven test selection process adapted to the needs and constraints of small-sized industrial applications, that comprises
 - A modelling language for the analysis phase based on UML2 for its concrete syntax with semantics given in terms of a formal language
 - A test selection language based on model transformations
 - That defines a set of test selection instructions specifically applicable to the targeted systems and based on industry best practices
 - A set of metrics that will help evaluating the test selection



The Four SESAME Artifacts

Analysis Model (FOREST)

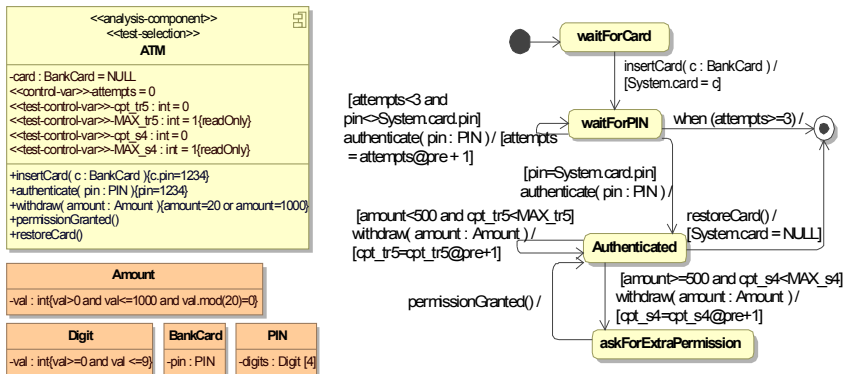


Test Selection Specification (SERELA)

```

select-val insertCard c.pin=1234
select-val authenticate pin=1234
select-val withdraw amount=20 or amount=1000
tr-repet t5 min 1 max 1
st-repet askForExtraPermission min 1 max 1
    
```

Test Model (FOREST)



Test Selection Metrics

State coverage (4/4)
 Transition coverage (6/8)
 Event coverage (4/4)
 Datatype coverage
 withdraw::amount (2/50)
 authenticate::pin (1/10000)
 ...

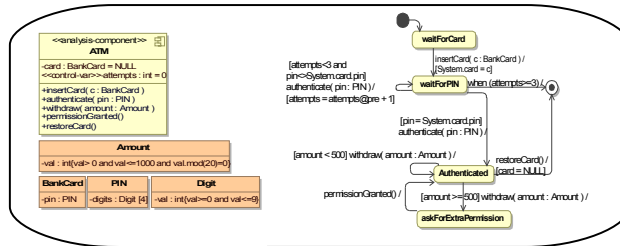
The Four SESAME Activities

Req.1) maximum of 3 bad authentications
 Req.2) withdraw max of 1000€
 Req.3) withdraw min of 20€
 ...

Software Requirements

1 Software Requirements Analysis

Analysis Model



datatype domain coverage

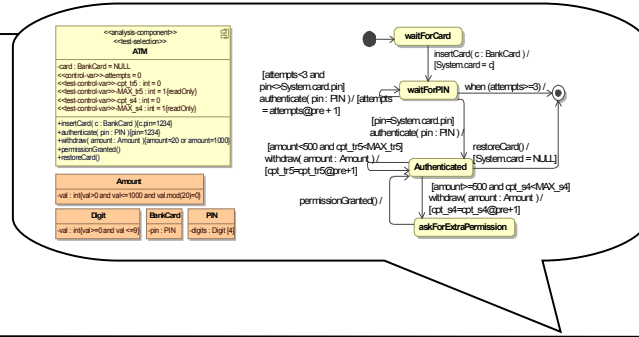
Test Selection Metrics

Test Requirements
 test boundary values for withdraw

2 Test Constraints Specification

Test Selection Specification

select-val withdraw amount=20 or amount=1000



Test Model

3 Test Selection Evaluation

Test Selection Coverage

amount datatype coverage : 2 out of 50

4 Test Generation

Abstract Test Cases

Test Case 01:
 insertCard(c.pin=1234);
 authenticate(9999);
 authenticate(1234);
 withdraw(20);
 restoreCard()

Model-driven Testing Process

Related Work

- *Model-driven testing using UTP* [Baker et al. 08]
 - + Notation is UML2 testing profile (UTP) [OMG05]
 - - UML2 interactions diagrams are used for specifying behaviors test selection → enumeration of test cases
- *Practical model-based testing* [Utting et al. 07]
 - + Comprehensive approach to model-based testing
 - - Test model manually created from models of the SUT or from the SUT itself with guidelines
 - - Extensive set of test selection criteria but no test selection language to specify them.
- *TestEra: Specification-based testing of Java programs using SAT* [Kurshid et al. 04]
 - + Formal test generation based on constraint solving technique (Alloy)
 - - Formal “analysis” model: expected system behavior is specified as pre-/post conditions in Alloy.

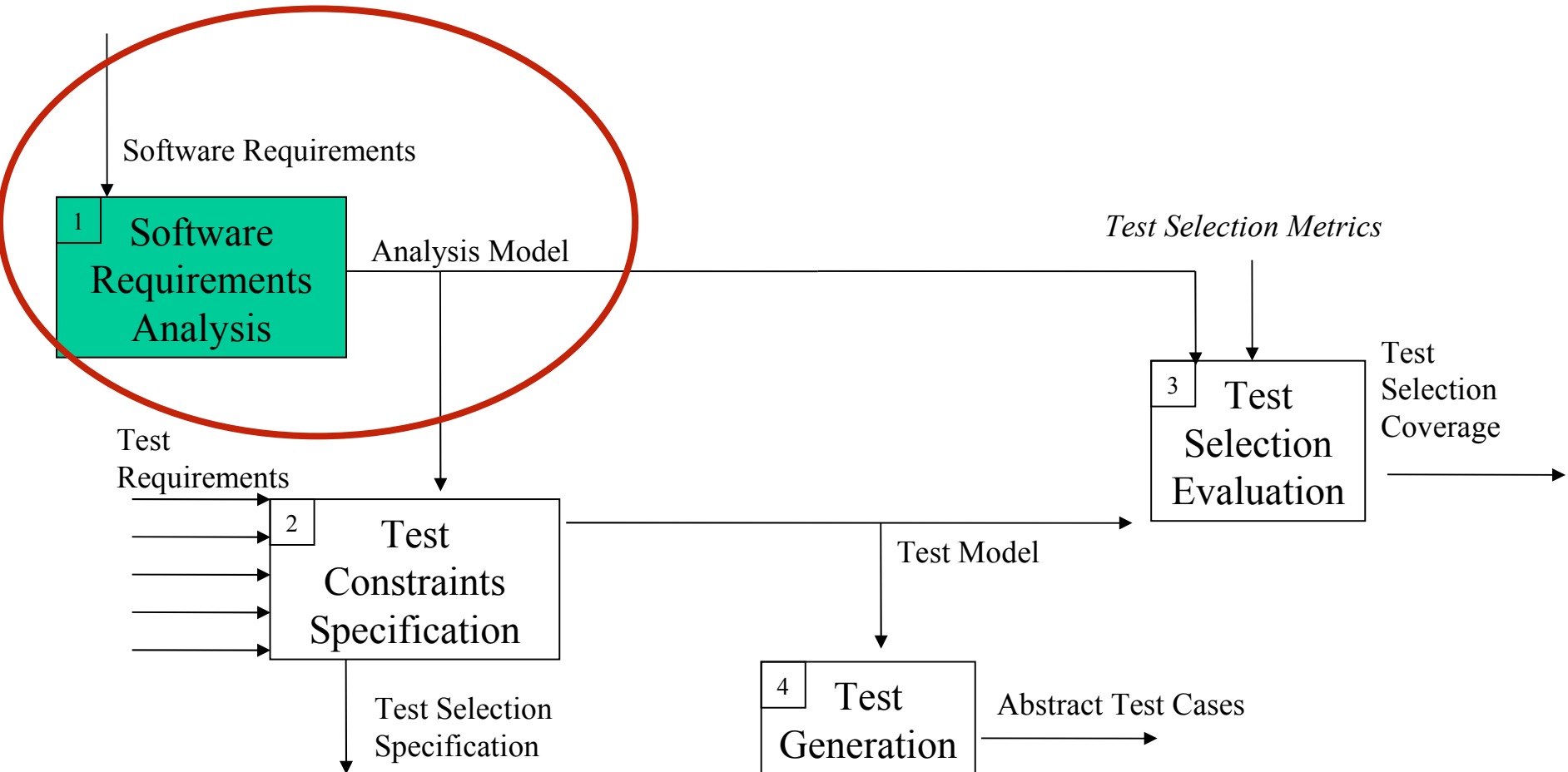


Outline

- Introduction
- Current Result: SESAME Process
 1. Analysis Phase
 2. Test Selection Specification
 3. Test Selection Evaluation
 4. Test Generation
- SESAME Finalization Tasks



The SESAME Test Selection Approach



SESAME Modeling Languages

- Two specification languages are needed
 - End-user modeling language (FOREST): usable and practical
 - FOREST language will be based on UML2 class diagrams and protocol state machines
 - Formal language: precise for test selection analysis/generation
 - Formal semantics of FOREST expressed in terms of Alloy models



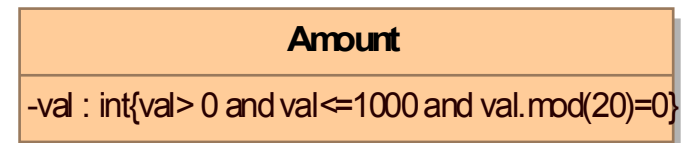
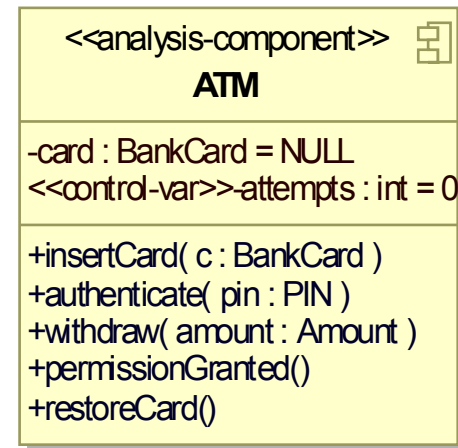
System Requirements Analysis

- Describe the interface of the system with its environment
 - Data definition (static view)
 - of the event parameters
 - of the visible state of the system
 - Allowed sequences of event receptions (dynamic view)



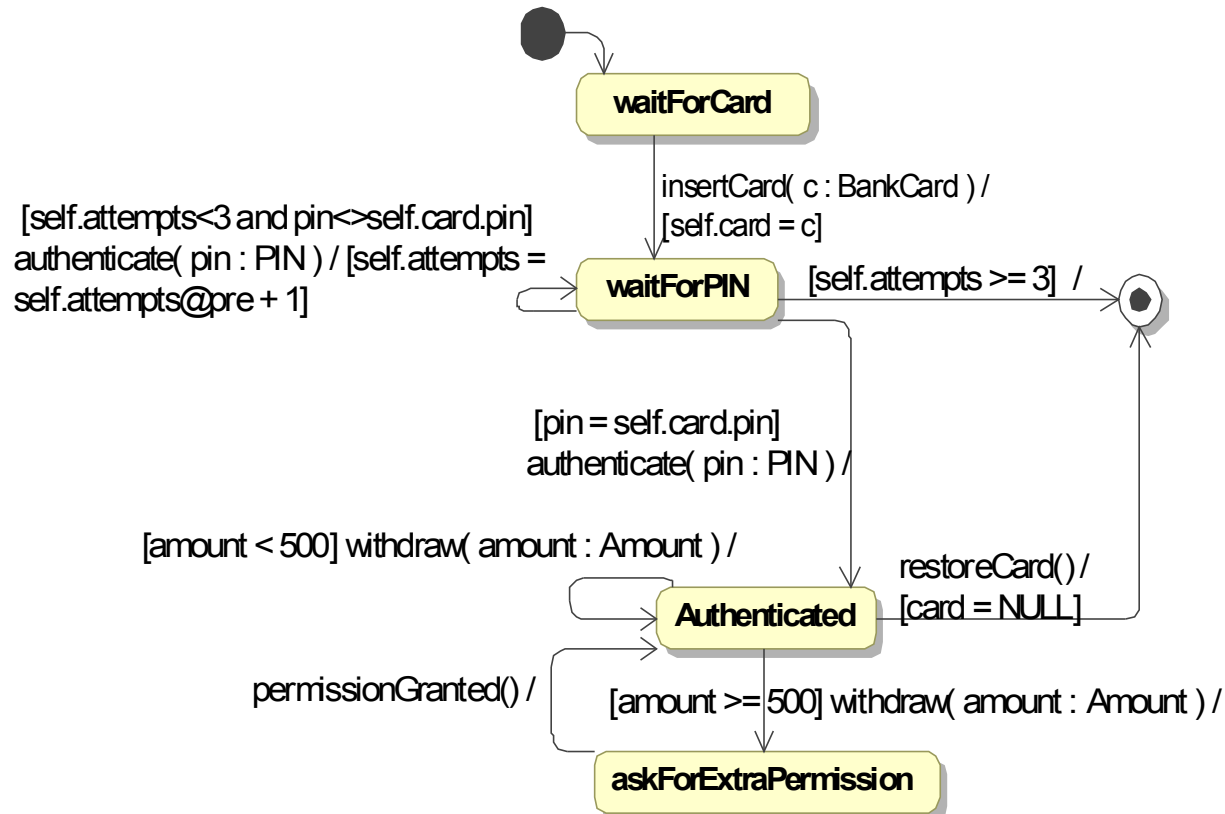
Analysis Model: Static View

- SUT component
- Events
- State variables
- Control variables
- Datatypes
- Basic data types (BDT) constraints

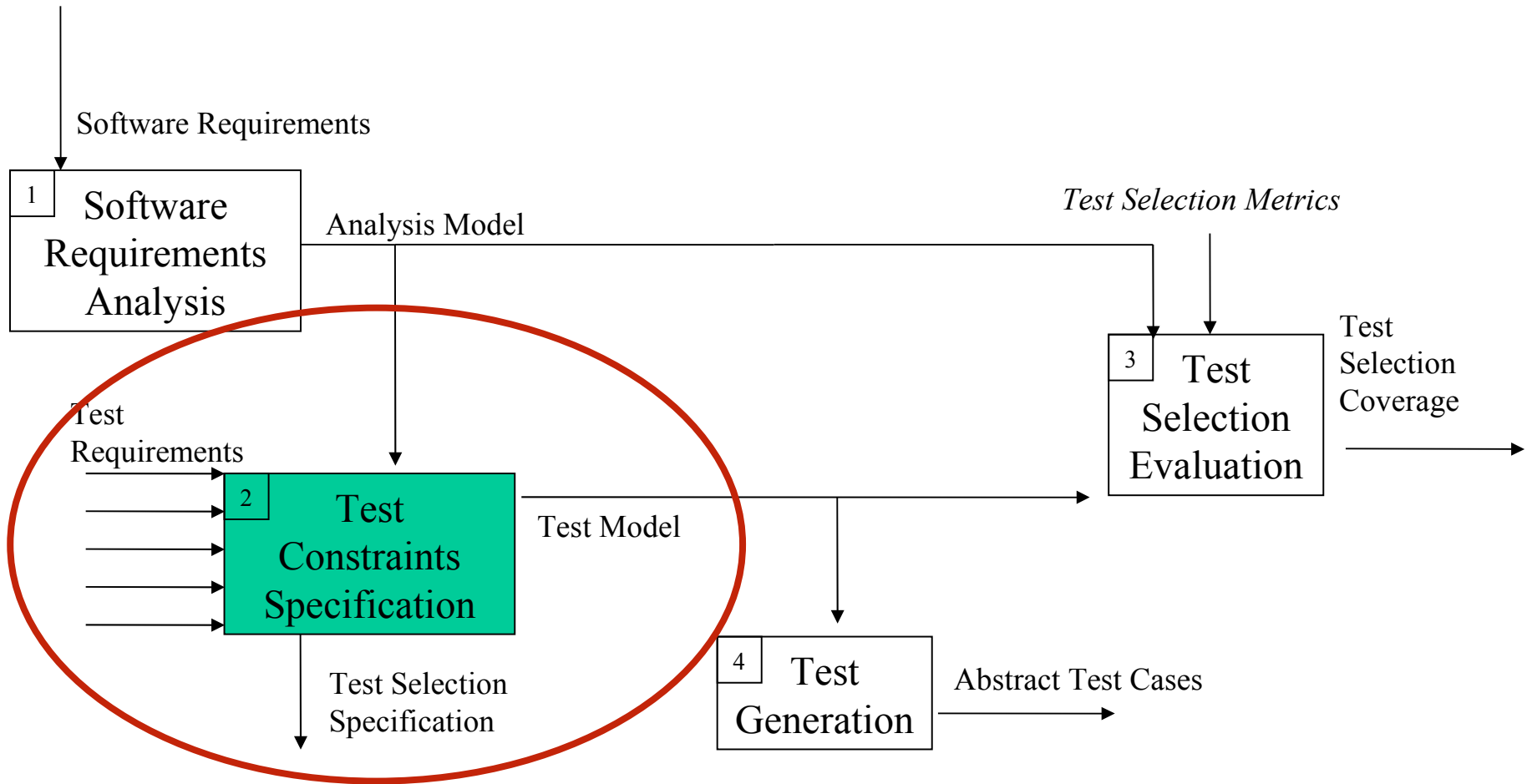


Analysis Model: Dynamic View

- Set of allowed sequences of system operation calls
- Pre and post conditions for each operation at each transition



The SESAME Test Selection Approach

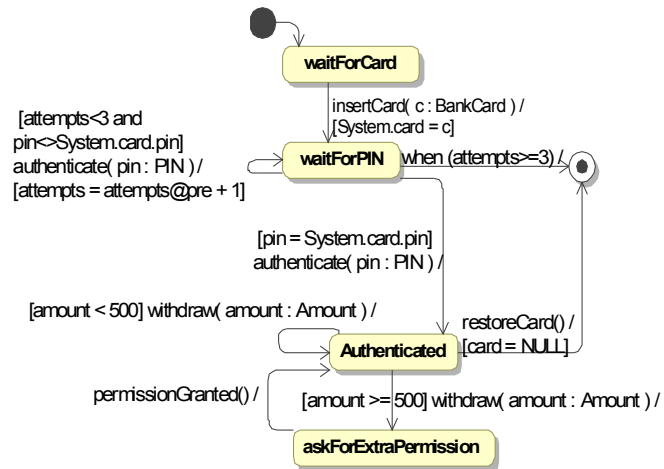
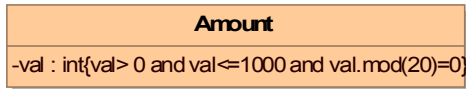
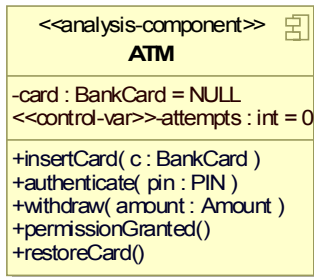


SERELA Test Selection Language

- Five basic test selection instructions have been defined:
 - Select value(s) of an event's parameters
 - `select-val EVT-NAME EXP`
 - Select value(s) of an event's parameters for a specific transition
 - `select-val-trans TRANS-NAME EXP`
 - Constrain the number of repetition of a transition, state, operation
 - `evt-repet EVT-NAME min NAT1 max NAT2`
 - `tr-repet TR-NAME min NAT1 max NAT2`
 - `st-repet S-NAME min NAT1 max NAT2`



Analysis Model



Test Selection Specification:

- Insert card with pin 1234

```
select-val insertCard c.pin=1234
```

- Authenticate with 1234 (good pin)

```
select-val authenticate pin=1234
```

- Withdraw 20 or 1000 euros

```
select-val withdraw amount=20 or amount=1000
```

- “cheap” withdraw event occurs exactly once

```
tr-repet t5 min 1 max 1
```

- “expensive” withdraw event occurs exactly once

```
st-repet askForExtraPermission min 1 max 1
```



SERELA Test Selection Language

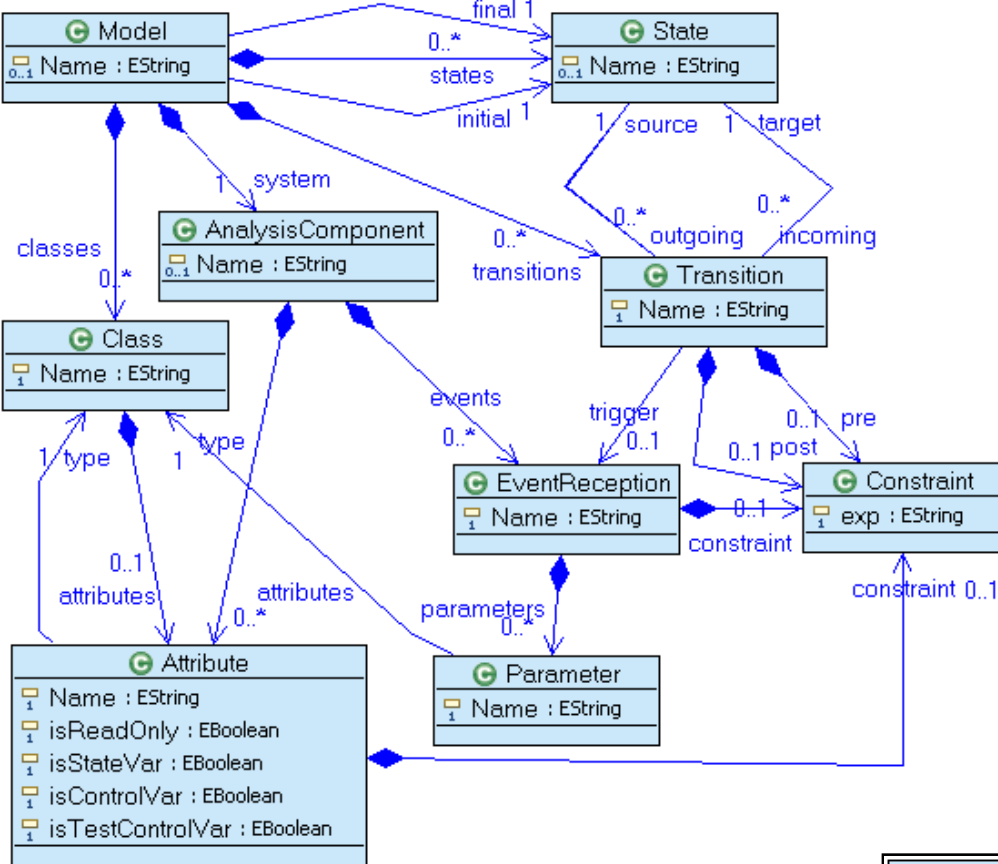
- Automotive-related test requirements
 - every software system embedded in a car must be able to switch to a sleep mode (AUTOSAR)
 - Maximum time before system wake-up
 - Temperature range values wrt safety standards



Test Selection Semantics: Tool-support

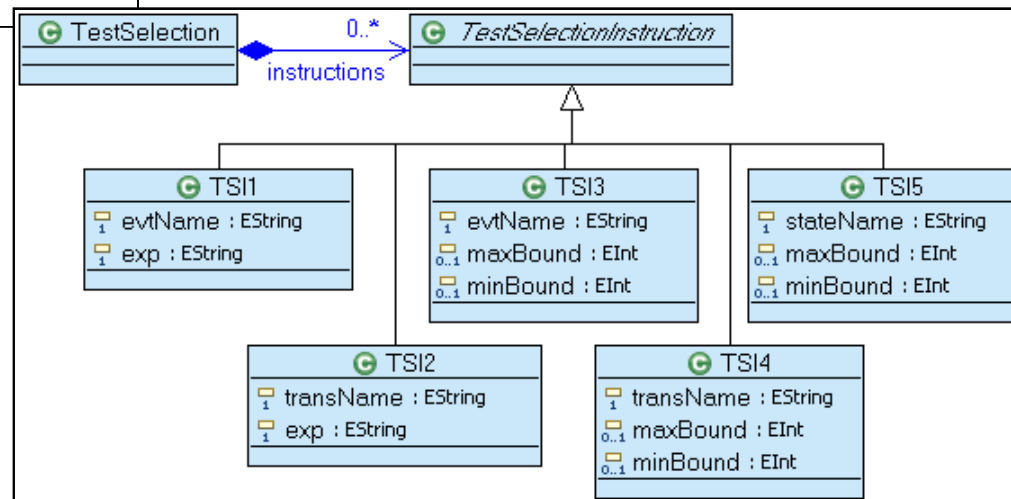
- Requirements
 - XMI-compliant model transformation engine
- Selection
 - Kermeta metamodeling tool
- Required Developments
 - FOREST metamodel definition
 - SERELA metamodel definition
 - SERELA test selection instruction implementation in Kermeta





SERELA Metamodel

FOREST Metamodel



Test Selection Semantics Implementation in Kermeta

```
operation main is do
  var analysisModel : Model
  var testModel : Model init Model.new
  var testSelection : TestSelection

  //[skipped] loading analysis model and metamodel from XMI files and
  initialize test model to analysis model

  testSelection.instructions.each{ tsi |
do
    if (tsi.isKindOf(TSI1)) then
      modelTransfo1(t.asType(TSI1), testModel)
    end
    if (tsi.isKindOf(TSI2)) then
      modelTransfo2(t.asType(TSI2), testModel)
    end
    //[skipped] same for other test constraints
  end
}

operation modelTransfo1(tsi: TSI1, model : Model) is do
  model.system.events.each{ evt |
do
    if (evt.Name.equals(tsi.evtName)) then
      evt.exp := evt.exp + " and (" + tsi.exp + ")"
    end
  end
end

//[skipped] implementation of other test constraints
```

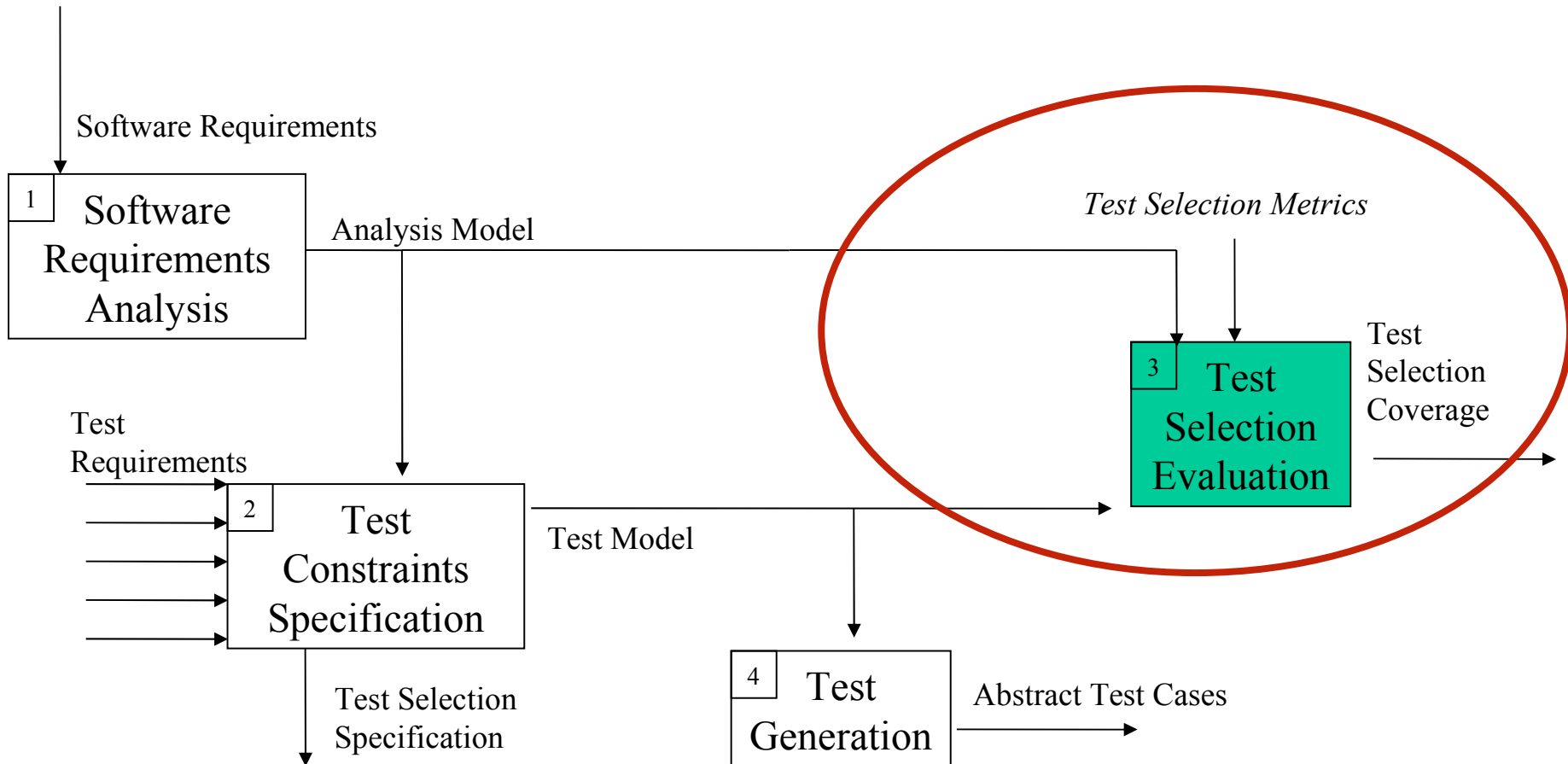


Test Selection Semantics: Contribution

- Definition of the metamodels in Kermeta
 - Analysis model (FOREST)
 - Test selection language (SERELA)
- Complete operational semantics of SERELA in Kermeta



The SESAME Test Selection Approach



Test Selection Evaluation: Introduction

- Comparison of values of the following metrics on the semantics of the analysis model and on the test models
 - State coverage
 - Transition coverage
 - Event coverage
 - Multiple (pre)conditions coverage
 - Data definition domain size of a given datatype
 - Maximum number of repetition of a given event



Test Selection Evaluation: Illustration

Metrics	Analysis model	Test model
Number of states	4	4
Number of transitions	8	6
Number of system operations	4	4
Maximum number of repetition of a given event		
insertCard	1	1
authenticate	3	1
withdraw	infinite	2
restoreCard	1	1
Data definition domain size		
withdraw::amount	50	2
authenticate::pin	10 000	1
insertCard::c	10 000	1
Multiple Conditions Coverage		
attempts < 3	Covered	Covered
attempts >= 3	Covered	Not covered
pin ≠ system.card.pin	Covered	Not covered
pin = system.card.pin	Covered	Covered
amount < 500	Covered	Covered
amount >= 500	Covered	Covered

```

<<analysis-component>>
<<test-selection>>
ATM

-card : BankCard = NULL
<<control-var>>attempts = 0
<<test-control-var>>cpt_tr5 : int = 0
<<test-control-var>>MAX_tr5 : int = 1{readOnly}
<<test-control-var>>cpt_s4 : int = 0
<<test-control-var>>MAX_s4 : int = 1{readOnly}

+insertCard( c : BankCard )(c.pin=1234)
+authenticate( pin : PIN )(pin=1234)
+withdraw( amount : Amount )(amount=20 or amount=1000)
+permissionGranted()
+restoreCard()
    
```

```

Amount
-val : int{val>0 and val<=1000 and val.mod(20)=0}
    
```

```

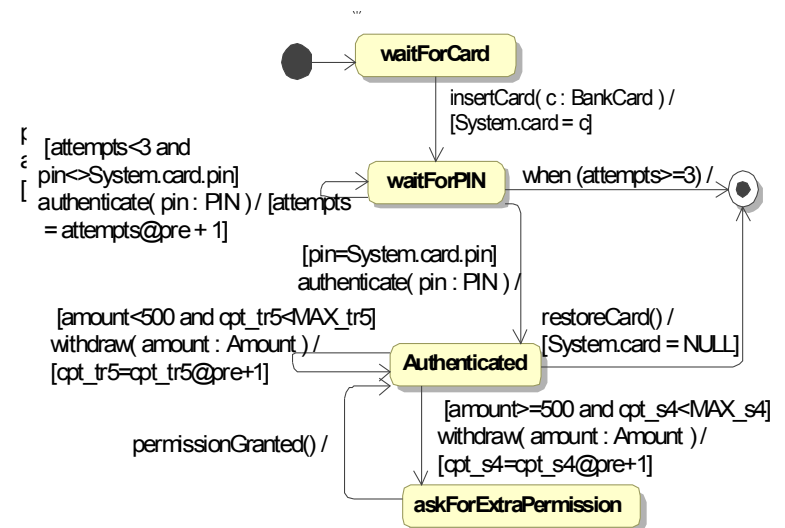
Digit
-val : int{val>=0 and val <=9}
    
```

```

BankCard
-pin : PIN
    
```

```

PIN
-digits : Digit [4]
    
```



Test Selection Evaluation: Introduction

- Comparison of values of the following metrics on the semantics of the analysis model and on the test models
 - States coverage
 - Transitions coverage
 - Events coverage
 - Multiple condition coverage
 - Data definition domain size of a given datatype
 - Maximum number of repetition of a given event
- In order to compute these metrics, the semantics of the analysis and test models (FOREST models) must be formally defined.



UML Semantics Variation Points

- UML state machines semantics variation points concerns the management of events, transitions and time [Chauvel et al.05]
 - FOREST event management
 - Select randomly one of the “available” events w.r.t. to current state and current state’s outgoing transitions’ preconditions
 - At a particular moment in time, at most one event can occur
 - FOREST transition management
 - If more than one transition can be fired by the selected event (non-determinism) then one transition is chosen randomly
 - FOREST time management
 - Synchrony hypothesis: Time is continuous. Events are processed as soon as they are received.



Alloy Overview

- Alloy is a structural modeling language based on first-order logic, for expressing complex structural constraints and behavior.
- Alloy Analyzer is a constraint solver that provides fully automatic simulation and checking. It is a 'model finder':
 - Given a logical formula (in Alloy), it attempts to find a model -- a binding of the variables to values -- that makes the formula true.
- Alloy has been developed by the Software Design Group at MIT. The first Alloy prototype was out a decade ago.
- In the SESAME process, the Alloy Analyzer is used to
 1. compute test selection metrics
 2. generate set of abstract test cases



FOREST Semantics in Alloy: Metamodel

```
sig SUT {  
  eventReceived : one Event,  
  curState : one State
```

```
}  
abstract sig Event{}  
abstract sig DataType{}
```

```
one sig PSM {  
  t : State -> State
```

```
}  
abstract sig State{}
```

Structure

```
sutOrd/first.curState = s1
```

```
fact step {  
  all sut : ATM - sutOrd/last | let sut' = sutOrd/next[sut] {  
    sut'.curState in successorStates[sut]  
    postcond[sut, sut']  
  }  
}
```

```
}  
fun successorStates (sut: ATM) : set State {  
  {suc : State | (sut.curState)->suc in PSM.t and precondition[sut, suc]}  
}
```

Execution

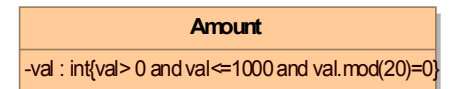
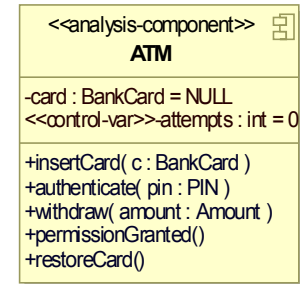


FOREST Semantics in Alloy: Model – Static View

- Analysis Component
sig ATM extends SUT {
 card : lone BankCard,
 attempts : Int
}

- Events
one sig restoreCard, permissionGranted, noEvent extends Event {}
sig withdraw extends Event {
 amount : Amount
}

- Datatypes
one sig BankCard extends DataType {pin : PIN}
sig Amount extends DataType {val : Int} { val > 0 and val < 50}

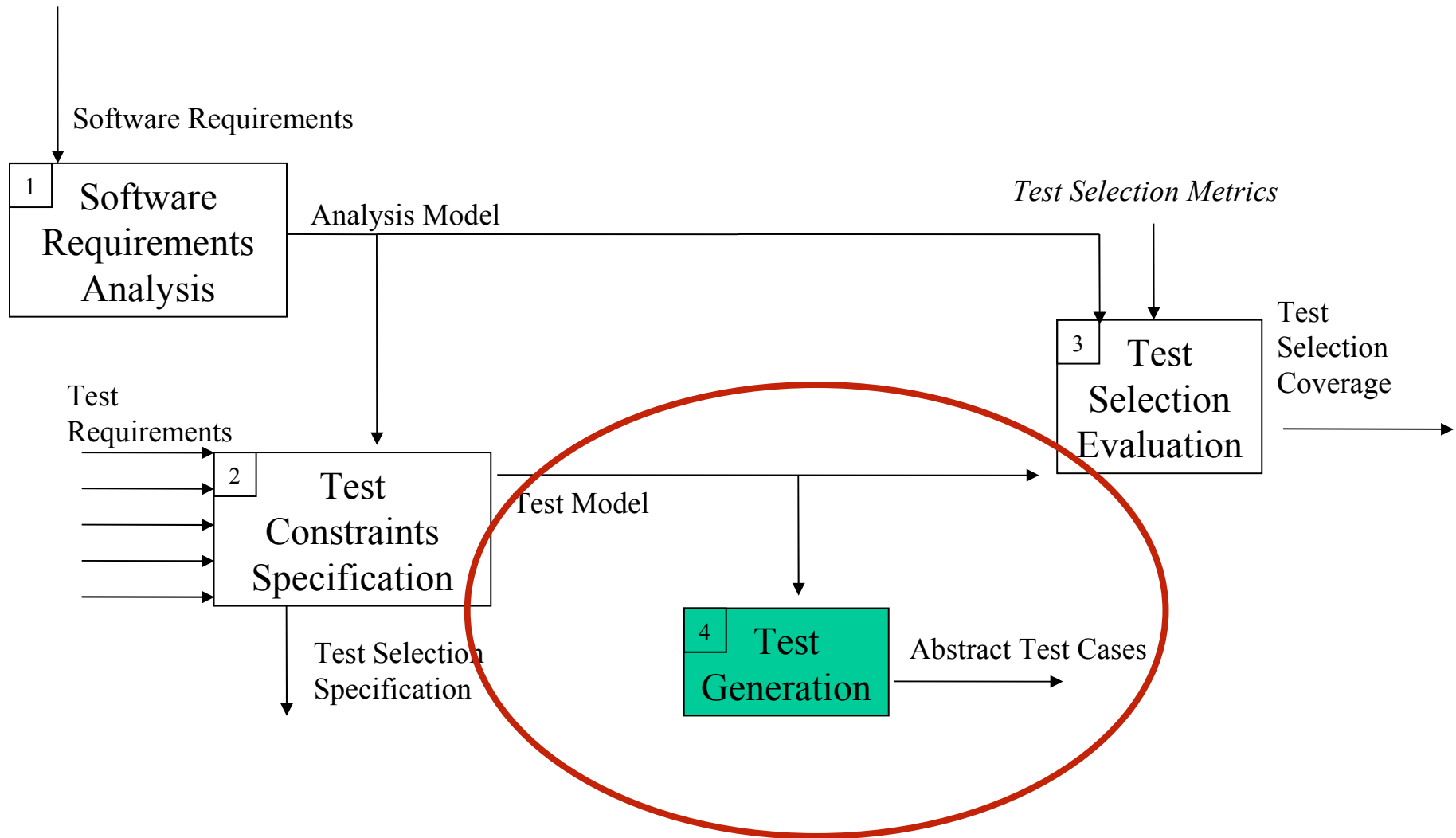


Test Selection Metrics in Alloy

- States coverage
 - run {s2 in sutOrd/nexsts[sutOrd/first].curState} for 10 but 5 int
 - run {s3 in sutOrd/nexsts[sutOrd/first].curState} for 10 but 5 int
 - run {s4 in sutOrd/nexsts[sutOrd/first].curState} for 10 but 5 int
- Events coverage
 - run {some sut : SUT | sut.eventReceived in insertCard} for 10 but 5 int
 - run {some sut : SUT | sut.eventReceived in authenticate} for 10 but 5 int
 - run {some sut : SUT | sut.eventReceived in withdraw} for 10 but 5 int
 - run {some sut : SUT | sut.eventReceived in permissionGranted} for 10 but 5 int
 - run {some sut : SUT | sut.eventReceived in restoreCard} for 10 but 5 int
- Transitions coverage
 - run {some sut, sut' : SUT | sut.curState = s1 and sut'.curState = s2} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s2 and sut'.curState = final} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s2 and sut'.curState = s2} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s2 and sut'.curState = s3} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s3 and sut'.curState = s3} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s3 and sut'.curState = s4} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s4 and sut'.curState = s3} for 10 but 5 int
 - run {some sut, sut' : SUT | sut.curState = s3 and sut'.curState = final} for 10 but 5 int



The SESAME Test Selection Approach



Test Generation with Alloy

- Objective: production of a set of abstract test cases
 - valid sequences of event occurrences with their parameters' values
- Solution: formalization of test models' semantics as an Alloy model enables the test generation automation
 - Running empty predicate on the Alloy test model ⇔ exhaustive test generation
 - `run {} for 10 but 5 int`



Outline

- Introduction
- Current Result: SESAME Process
- **SESAME Finalization Tasks**



SESAME Finalization Tasks

- Test Selection Specification Phase
 - Enrich the SERELA test selection language with instructions from IEE best practices
- Test Selection Evaluation Phase
 - Enrich the list of test selection metrics (e.g. requirements coverage, equivalence classes coverage)
- Experimental Validation of the SESAME Process on IEE Case Study
- Other finalization tasks for the analysis, test model construction and test selection phases mainly consisting of implementation of tool-support



References

- Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: UML2Alloy: A Challenging Model Transformation. In Engels, G., Opdyke, B., Schmidt, D., Weil, F., eds.: ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems. Volume 4735 of LNCS., Nashville, USA, Springer (2007) 436-450
- Baker, P., Dai, Z. R., Grabowski, J., Haugen, O., Schieferdecker, I., and Williams, C. 2008. Model-driven Testing: Using the UML Testing Profile. Springer.
- Khurshid S., and Marinov D. TestEra: Specification-based testing of Java programs using SAT. Automated Software Engineering Journal, 2004.
- Liuying L., and ZhiChang, Q. Test selection from UML statecharts, In Technology of Object-Oriented Languages and Systems (TOOLS). IEEE Comp. Society, 1999, 273-279.
- OMG. July 2005. UML 2.0 testing profile. Full Specification formal/05-07-07, Object Management Group.
- Utting, M. and Legeard, B. 2007. Practical Model-based Testing. Morgan Kaufmann, San Francisco.

