Runtime testing generated systems from Rebel specifications

Master Software Engineering

Thanusijan Tharumarajah

Amsterdam • 12 October 2017

UNIVERSITEIT VAN AMSTERDAM

ING

Agenda

1. Rebel background
2. Problem statement
3. Research question
4. Solution
5. Experiments
6. Most promising experiment
7. Demo
8. Conclusion

Rebel background

- Formal specification language (domain specific)
  - Easy to write and understand
  - No more missing or ambiguous specifications
- Financial industry
- Simulate, check and visualize specifications (Z3 solver)
  - Early fault detection
  - Reason about changes, errors and behaviour
- Code generation

Rebel background

```
specification Account {
  fields {
    accountNumber: IBAN @key
    balance: Money
  }

  events {
    openAccount[]
  }


  lifeCycle {
    initial init -> opened: openAccount
    final opened
  }
}



event openAccount[minimalDeposit: Money = EUR  0.00](initialDeposit: Money) {
  preconditions {
    initialDeposit >= minimalDeposit;
  }
  postconditions {
    new this.balance == initialDeposit;
  }
}
```

Problem statement

- Generated system leaves Rebel domain
  - Loss of testing and reasoning
  - Not tested

- Generated systems need to conform to the specifications

How to validate the generated code from a Rebel specification?

1. How is the input/output of the generated system tested?
2. Which false positives occur when the generated system is correctly implemented?
3. What kind of faults can be found and what are the factors?

ING

Solution

- SMT solver holds the key in testing
  - Checking
  - Simulation
- Testing at runtime
- Fault: The deviation between the current behaviour and the expected behaviour

Experiment: Invalid execution

- **Approach**
  - Mutation testing
- **Verification techniques**
  - Checking
- **Faults**
  - Two faults
  - Templating and compilation
- **Limitations**
  - Traces not used
  - Checking focuses on states, test framework on transitions

Experiment: Valid execution

- **Approach**
  - Model-based testing
- **Verification techniques**
  - Checking
  - Simulation
- **Faults**
  - Three faults
  - Templating and distribution
- **Limitations**
  - Not all transitions
  - Rebel interpretation in the SMT solver

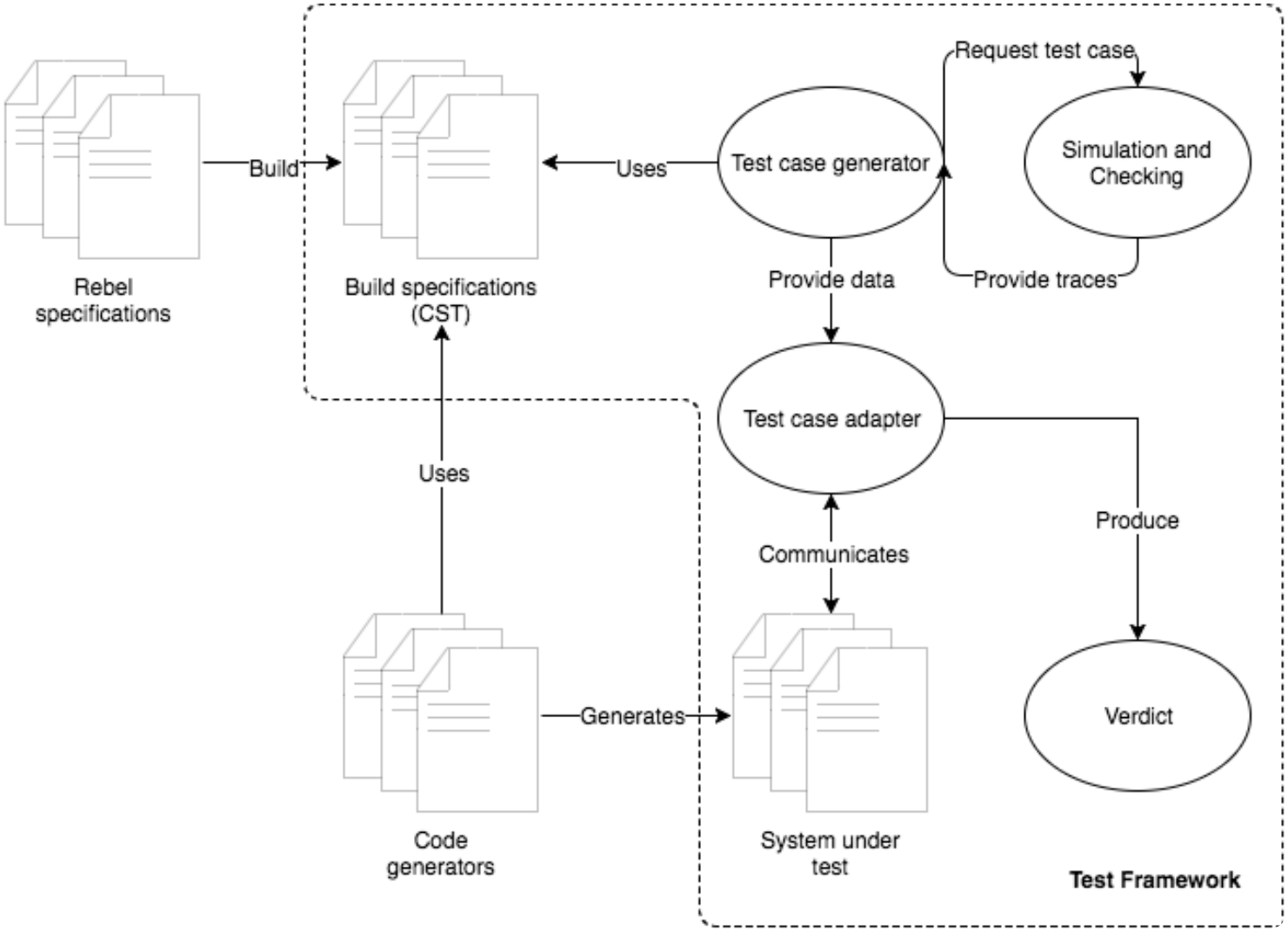
Most promising experiment: valid execution

Expectations beforehand

- Templating (error prone)
- Distribution
  - Partial failure
  - Asynchrony
- Efficiency

```
event book() {
  sync {
    Account[this.from].withdraw(this.amount);
    Account[this.to].deposit(this.amount);
  }
}
```

Most promising experiment: valid execution



ING

Most promising experiment: valid execution

Execute and test transition

- Pre-transition check
  - Current state
  - Current state values
- Transition check
  - Transition parameters data values
  - Traces
- Post-transition check
  - Test case adapter
  - Normalisation

Most promising experiment: valid execution

Codegen-Javadatomic

- Fault in interest transition
- Fault in templating
  - Fixed code

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = MD14FLBLJOYGVJMDUZVKLU4C

  instance: simple_transaction.Account, key = MD14FLBLJOYGVJMDUZVKLU4C
    State = opened
    ?
    var accountNumber (type: IBAN) = MD14FLBLJOYGVJMDUZVKLU4C
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/MD14FLBLJOYGVJMDUZVKLU4C/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"","isSuccessful":"false","message":"Bad Request",
           "errorBody":"","code":"400")
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = MD14FLBLJOYGVJMDUZVKLU4C

  instance: simple_transaction.Account, key = MD14FLBLJOYGVJMDUZVKLU4C
    State = opened
    ?
    var accountNumber (type: IBAN) = MD14FLBLJOYGVJMDUZVKLU4C
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/MD14FLBLJOYGVJMDUZVKLU4C/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"","isSuccessful":"false","message":"Bad Request",
           "errorBody":"","code":"400")
```

```
{
    "_id": 17592186045441,
    "_version": 1,
    "_status": "OPENED",
    "accountNumber": {
        "iban": "MD14FLBLJOYGVJMDUZVKLU4C"
    },
    "balance": {
        "value": 50.00,
        "currency": "EUR"
    }
}
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = MD14FLBLJOYGVJMDUZVKLU4C

  instance: simple_transaction.Account, key = MD14FLBLJOYGVJMDUZVKLU4C
    State = opened
    ?
    var accountNumber (type: IBAN) = MD14FLBLJOYGVJMDUZVKLU4C
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/MD14FLBLJOYGVJMDUZVKLU4C/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"","isSuccessful":"false","message":"Bad Request",
           "errorBody":"","code":"400")
```

```
{
    "_id": 17592186045441,
    "_version": 1,
    "_status": "OPENED",
    "accountNumber": {
        "iban": "MD14FLBLJOYGVJMDUZVKLU4C"
    },
    "balance": {
        "value": 50.00,
        "currency": "EUR"
    }
}
```

```
if(! (isLessOrEqualThan(currentInterest, 10 /* % */))) {
  throw new BuildCASTransactionException("Predicate did not hold: InterestTransaction: currentInterest
      <= 10%");
}
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = MD14FLBLJOYGVJMDUZVKLU4C

  instance: simple_transaction.Account, key = MD14FLBLJOYGVJMDUZVKLU4C
    State = opened
    ?
    var accountNumber (type: IBAN) = MD14FLBLJOYGVJMDUZVKLU4C
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/MD14FLBLJOYGVJMDUZVKLU4C/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"","isSuccessful":"false","message":"Bad Request",
           "errorBody":"","code":"400")
```

```
{
    "_id": 17592186045441,
    "_version": 1,
    "_status": "OPENED",
    "accountNumber": {
        "iban": "MD14FLBLJOYGVJMDUZVKLU4C"
    },
    "balance": {
        "value": 50.00,
        "currency": "EUR"
    }
}
```

```
if(! (isLessOrEqualThan(currentInterest, 10 /* % */))) {
  throw new BuildCASTransactionException("Predicate did not hold: InterestTransaction: currentInterest
      <= 10%");
}
```

```
public static boolean isLessOrEqualThan(BigDecimal lhs, BigDecimal rhs) {
  return lhs.compareTo(rhs) >= 0;
}
```

Most promising experiment: valid execution

Distributed Codegen-Akka

- Approach
  - Pre-transition and transition check to Node 1
  - Post-transition check from Node 2
- Test for all transitions fail
- Fault in distribution
  - Non unanimous final outcome
  - Serialisation error

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = FO1227539908389742

  instance: simple_transaction.Account, key = FO1227539908389742
    State = opened
    ?
    var accountNumber (type: IBAN) = FO1227539908389742
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/FO1227539908389742/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"CommandSuccess(Interest(-77.09))","isSuccessful":
          "true","message":"OK", "errorBody":"","code":"200")
Could not find state opened, expected  "state":{"Opened":{}}
Could not find value balance, expected  "balance":"EUR -3804.50"
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = FO1227539908389742

  instance: simple_transaction.Account, key = FO1227539908389742
    State = opened
    ?
    var accountNumber (type: IBAN) = FO1227539908389742
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/FO1227539908389742/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"CommandSuccess(Interest(-77.09))","isSuccessful":
          "true","message":"OK", "errorBody":"","code":"200")
Could not find state opened, expected  "state":{"Opened":{}}
Could not find value balance, expected  "balance":"EUR -3804.50"
```

```
Node 2
Endpoint: /Account/FO1227539908389742
Response:
("body":"","isSuccessful":"false","message":"Service Unavailable",
 "errorBody":"The server was not able to produce a timely response
 to your request.\r\nPlease try again in a short while!","code":"503")
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = FO1227539908389742

  instance: simple_transaction.Account, key = FO1227539908389742
    State = opened
    ?
    var accountNumber (type: IBAN) = FO1227539908389742
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/FO1227539908389742/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"CommandSuccess(Interest(-77.09))","isSuccessful":
           "true","message":"OK", "errorBody":"","code":"200")
Could not find state opened, expected  "state":{"Opened":{}}
Could not find value balance, expected  "balance":"EUR -3804.50"
```

```
Node 2
Endpoint: /Account/FO1227539908389742
Response:
("body":"","isSuccessful":"false","message":"Service Unavailable",
 "errorBody":"The server was not able to produce a timely response
 to your request.\r\nPlease try again in a short while!","code":"503")
```

```
Node 1
Endpoint: /Account/FO1227539908389742
Response:
{
   "state":{
      "SpecificationState":{
         "state":{
            "Opened":{

            }
         }
      }
   },
   "data":{
      "Initialised":{
         "data":{
            "accountNumber":null,
            "balance":"EUR -3804.50"
         }
      }
   }
}
```

```
2:
  now = 12 Jul 2016, 12:00:00
  step: simple_transaction.Account.interest
    var currentInterest (type: Percentage) = (- 7709)
    Transition to state = opened
    Identified by accountNumber = FO1227539908389742

  instance: simple_transaction.Account, key = FO1227539908389742
    State = opened
    ?
    var accountNumber (type: IBAN) = FO1227539908389742
    var balance (type: Money) = - EUR3804.50

Endpoint: /Account/FO1227539908389742/Interest
JSON payload: { "Interest": { "currentInterest":"-77.09" } }
Response: ("body":"CommandSuccess(Interest(-77.09))","isSuccessful":
           "true","message":"OK", "errorBody":"","code":"200")
Could not find state opened, expected  "state":{"Opened":{}}
Could not find value balance, expected  "balance":"EUR -3804.50"
```

```
Node 2
Endpoint: /Account/FO1227539908389742
Response:
("body":"","isSuccessful":"false","message":"Service Unavailable",
 "errorBody":"The server was not able to produce a timely response
 to your request.\r\nPlease try again in a short while!","code":"503")
```

```
Node 1
Endpoint: /Account/FO1227539908389742
Response:
{
   "state":{
      "SpecificationState":{
         "state":{
            "Opened":{

            }
         }
      }
   },
   "data":{
      "Initialised":{
         "data":{
            "accountNumber":null,
            "balance":"EUR -3804.50"
         }
      }
   }
}
```

```
[info] [ERROR] [09/14/2017 14:42:31.146] Failed to serialize remote message
  [class com.ing.rebel.Rebel$CurrentState$CurrentStateInternal] using serializer
    [class com.romix.akka.serialization.kryo.KryoSerializer].
  Transient association error (association remains live)
[info] akka.remote.MessageSerializer$SerializationException: Failed to serialize remote message
  [class com.ing.rebel.Rebel$CurrentState$CurrentStateInternal] using serializer
  [class com.romix.akka.serialization.kryo.KryoSerializer].
```

Demo

Conclusion

- **Testing approaches**
  - Valid execution
  - Invalid execution

- **Fault categories**
  - Templating
    - Fixed code
    - Injected code
  - Compilation
  - Distribution

Future work

- Mutation model-based testing
- Bounded checking
- Invalid current state
- More complex specifications

Reflection

- Master's thesis @ ING
- Rebel project

Thank you

ING