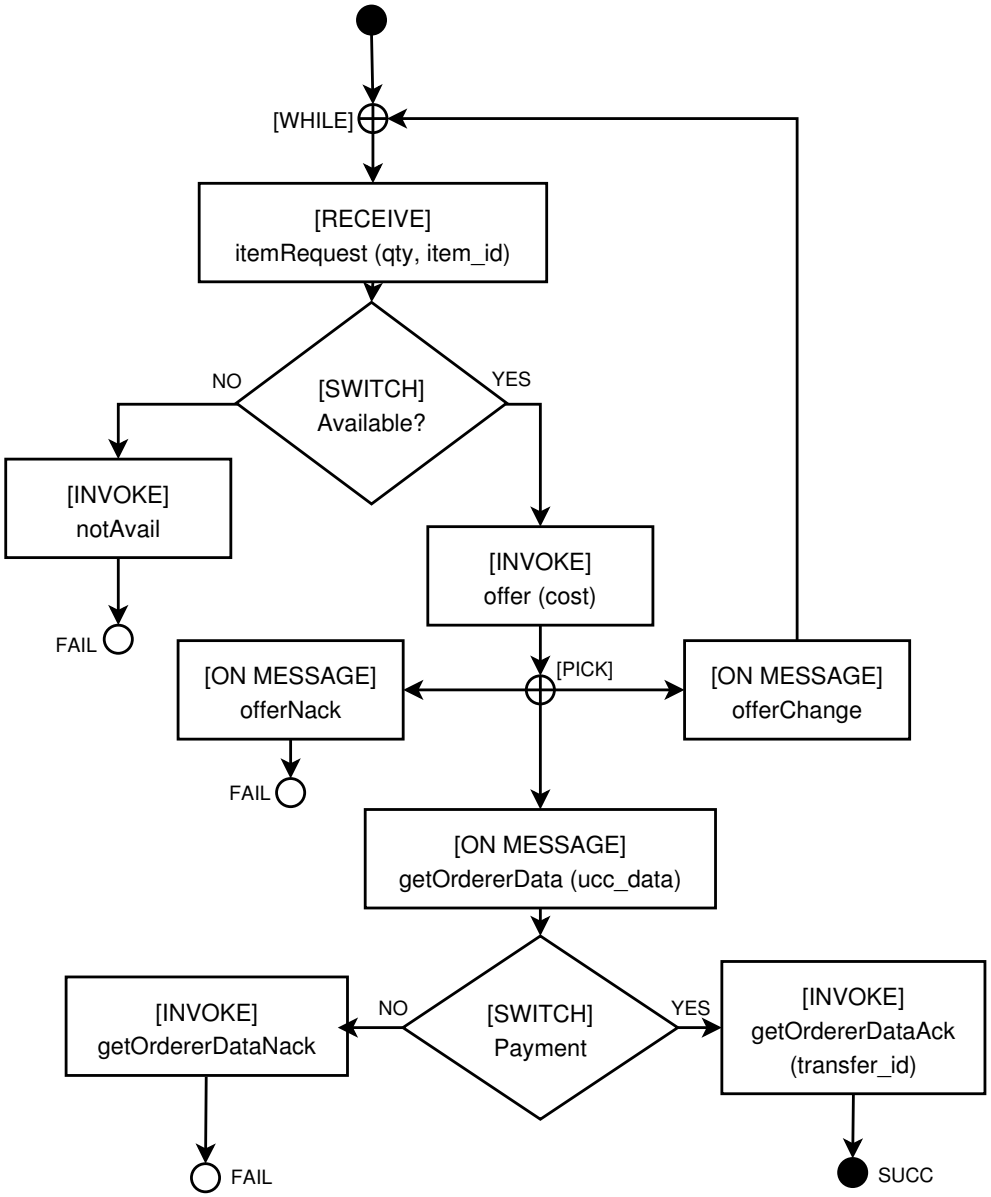
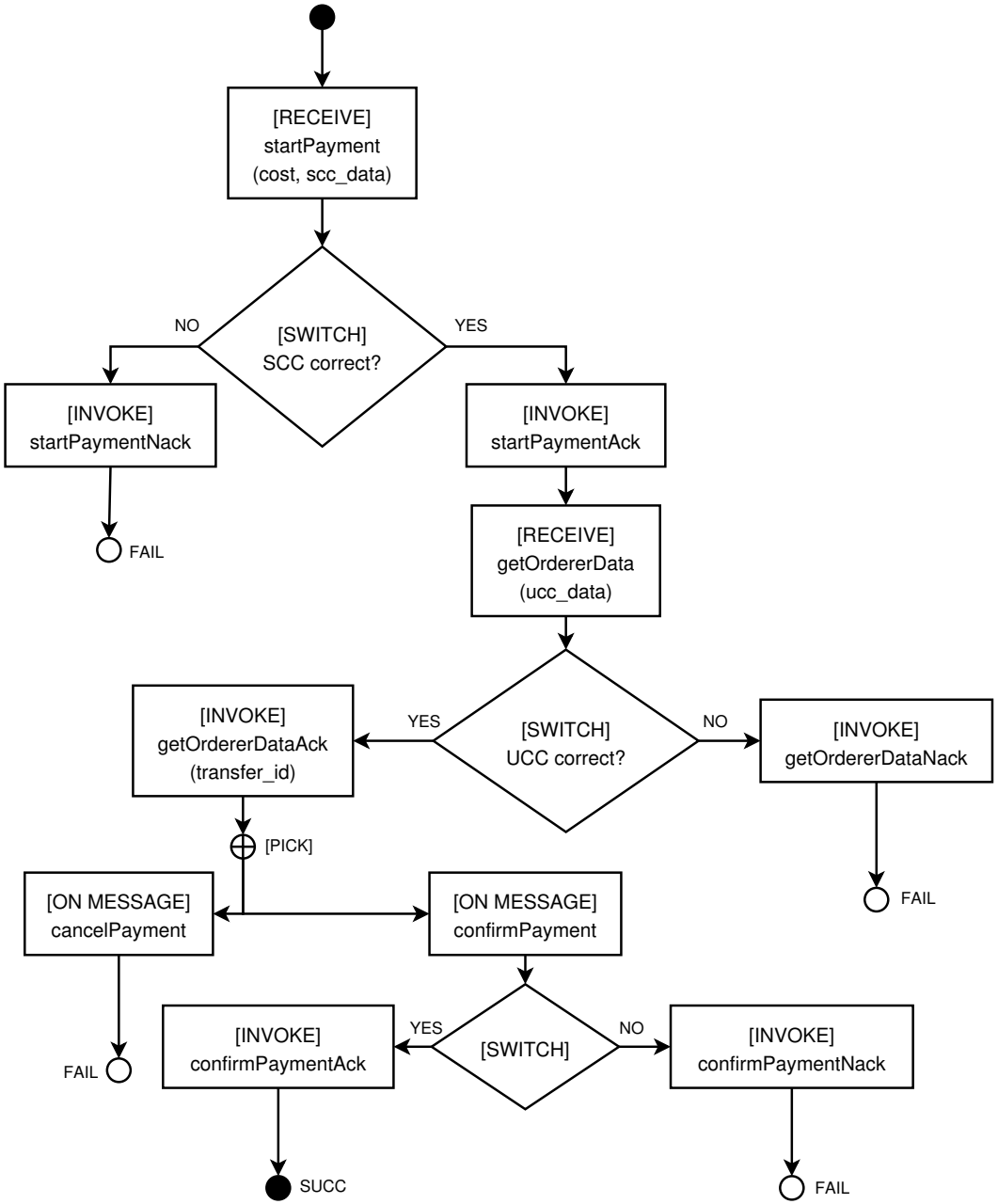
RTML formulas. Talk outline

- VOS (*Virtual Online Shop*) example
- Atoms (Language of events)
- Instance monitor formulas
- Class monitor formulas
- Translation to Java code

VOS Example: VOS Abstract



VOS Example: Bank



Virtual Online Shop example

1. **RetriesOnSuccCount**: count the number of items offered to the User before the User accepts to buy.
2. **PaymentTime**: compute the time requested to finalize the payment with the bank from the payment request.
3. **GlobalStoreCcNotRefuse**: the credentials of the Store have never been refused by the Bank in any execution of the VOS.
4. **CountStoreCCRefused**: count the total number of times the Bank has refused the credentials of the Store on all the executions of the VOS.
5. **AverageUserRetriesCount**: average number of times the user gets and refuses an offer from the VOS.
6. **AveragePaymentTime**:average duration of the interactions with the back for the payment procedure.

Language atoms (events)

$$e ::= \textit{link}.\text{start} \mid \textit{link}.\text{end} \mid$$
$$\text{msg}(\textit{link}.\text{input/output} = \textit{msg}[\textit{opt-constraints}]) \mid$$
$$\text{cause}(\textit{link}.\textit{var} = \textit{val}) \mid$$
$$\text{cause}(\textit{link}.\text{state} = \textit{label})$$

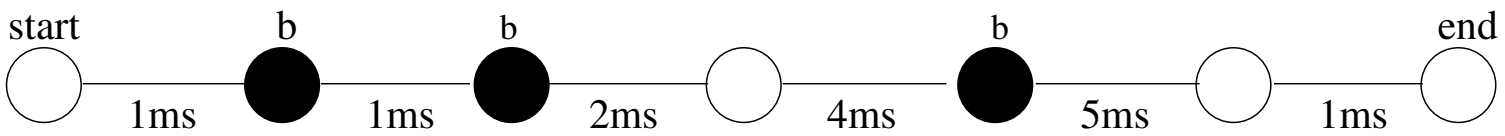
Examples:

- `msg(Bank.input = startPayment)`
- `msg(Shop.input = request [item = 'Book'])`
- `cause(Bank.state = FAIL)`
- `cause(Bank.startPayment_cost = e100)`

Instance monitor formulas

$$
\begin{aligned}
b ::= {} & e \mid \mathsf{Y}\, b \mid \mathsf{O}\, b \mid \mathsf{H}\, b \mid b\ \mathsf{S}\ b \mid \\
& n = n \mid n > n \mid \neg b \mid b \wedge b \mid \text{true} \\
n ::= {} & \text{count}\,(b) \mid \text{time}\,(b) \mid b?n : n \mid \\
& n + n \mid n - n \mid\ n * n \mid n/n \mid 0 \mid 1 \mid \ldots
\end{aligned}
$$

- $\mathsf{Y}\, b$ means "b was true in the previous step";
- $\mathsf{O}\, b$ means "b was true (at least) once in the past";
- $\mathsf{H}\, b$ means "b was true always in the past";
- $b_1 \mathsf{S}\, b_2$ means "b_1 has been true since b_2".



Instance monitor examples

- **RetriesOnSuccCount**:
  $\mathsf{O}\,(\mathtt{cause}(\mathtt{VOS.state} = \mathtt{SUCC}))?$
  $\mathsf{count}\,(\mathtt{msg}(\mathtt{VOS.ouput} = \mathtt{offer}))) : 0$
- **PaymentTime**:
  $\mathsf{time}\,((\neg(\mathtt{cause}(\mathtt{Bank.state} = \mathtt{SUCC}, \mathtt{FAIL}))\mathsf{S}$
  $\mathtt{msg}(\mathtt{Bank.input} = \mathtt{startPayment})))$

Class monitor formulas

$$
\begin{aligned}
B ::= & \text{And}\,(b) \mid \text{Y}\,B \mid \text{O}\,B \mid \text{H}\,B \mid B\ \text{S}\,B \mid \\
& N = N \mid N > N \mid \neg B \mid B \wedge B \mid \text{true} \\
N ::= & \text{Count}\,(b) \mid \text{Sum}\,(n) \mid \\
& N + N \mid N - N \mid\ N * N \mid N/N \mid 0 \mid 1 \mid \ldots
\end{aligned}
$$

(“Average (n)” abbreviates “Sum (n)/Count $(\text{O}\,\text{start})$”)

Class monitor examples

- **GlobalStoreCcNotRefuse**:
  And $(\neg \mathsf{O}\, \mathtt{msg}(\mathtt{Store.input} = \mathtt{startPaymentNack}))$
- **CountStoreCCRefused**:
  Count $(\mathsf{O}\, \mathtt{msg}(\mathtt{Store.input} = \mathtt{startPaymentNack}))$
- **AverageUserRetriesCount**:
  Average $(\text{count}\, (\mathtt{msg}(\mathtt{VOS.output} = \mathtt{offer})))$
- **AveragePaymentTime**:
  Average $(\text{time}\, ((\neg(\mathtt{cause}(\mathtt{Bank.state} = \mathtt{SUCC}, \mathtt{FAIL}))$
  $\mathsf{S}\, \mathtt{msg}(\mathtt{Bank.input} = \mathtt{startPayment}))))$

RTML to Java for instance monitors

Keep in a map *Val* the (numerical or boolean) value for each sub-formula. On events, update *Val* bottom-up on the structure of the formula.

- $Val(e) :=$ "if event e is occurring"
- $Val(b_1 \wedge b_2) := (Val(b_1) \wedge Val(b_2))$
- $Val(s_1 = s_2) := (Val(s_1) = Val(s_2))$
- $Val(\mathrm{Y}\, b) := Val_{old}(b)$
- $Val(\mathrm{O}\, b) := Val_{old}(\mathrm{O}\, b) \vee Val(b)$
- $Val(\mathrm{count}\,(b)) :=$ if $Val(b)$ then $(Val_{old}(\mathrm{count}\,(b)) + 1)$ else $Val_{old}(\mathrm{count}\,(b))$
- $Val(\mathrm{time}\,(b)) :=$ if $Val(b) \wedge Val_{old}(b)$ then $(Val_{old}(\mathrm{time}\,(b)) + \mathrm{elapsed})$ else $Val_{old}(\mathrm{time}\,(b))$
- ...

RTML to Java for class monitors

1. An instance monitor for every sub-formula “under” And , Count and Sum
2. An aggregating monitor built with same idea of map *Val* for instance monitor. For terminal sub-formula:

- $Val(\text{And}(b)) := \bigwedge_{i \in \mathcal{I}(b)} i.Val(b)$
- $Val(\text{Count}(b)) := \sum_{i \in \mathcal{I}(b)} (\text{if } i.Val(b) \text{ then 1 else 0})$
- $Val(\text{Sum}(n)) := \sum_{i \in \mathcal{I}(n)} i.Val(n)$

RTML to java example: **AverageUserRetriesCount**

```
package monitor.instance;
import org.astroproject.monitor.core.*;
import org.astroproject.monitor.utility.Utility;
public class Class_VOS_GlobStoreRefuseCc implements IProcessClassMonitor, IStatis
ticPropertyMonitor {
    // Formula: "COUNT  O KNOW Store.state = pc234"
    public int status() { return IMonitor.STATUS_RUNNING; }
    public void init() { }
    public void update() {
        state[0]=Utility.computeCount("monitor.instance.VOS__GlobStoreRefuseCc_0");
    }
    float state[] = new float[1];
    public float value() { return state[0]; }
    public String getErrorNode() { return "no error"; }
    public String getProcessName() { return "VOS"; }
    public String getProperty() { return "GlobStoreRefuseCc"; }
    public String getDescription() { return "Store credentials are never refused by bank"; }
}
```

RTML to java example: **AverageUserRetriesCount**

```
package monitor.instance;
import org.astroproject.monitor.core.*;
import java.util.ArrayList;
import java.lang.Integer;
public class VOS__GlobStoreRefuseCc_0 implements IProcessInstanceMonitor, IServan
tMonitor, IBooleanPropertyMonitor {
    private float state[] = new float[3];
    private boolean is_valid = true;
    public void evolve(BpelMsg msg)
    {
        if(msg.getSenderType().indexOf("Store")==0 ||
           msg.getReceiverType().indexOf("Store")==0) {
            float next[] = new float[3];
            next[1]=(msg.getOperation.equals("startPaymentNack"))?1:0;
            next[2]=((next[1]==1)||(state[2]==1))?1:0;
            is_valid=(next[2]==1);
            state=next;
        }
        return;
    }
    void init()
    {
        state[1]=(Store.p1())?1:0;
        state[2]=state[1];
        is_valid=(state[2]==1);
        return;
    }
    public boolean isValid() { return is_valid; }
}
```