SRML and Policies

Laura Bocchi
bocchi@mcs.le.ac.uk
Agenda

- Internal configuration policies
- External configuration policies
- Non-functional properties and SLA
A **SRML module** describes one instance of the session of a **service** or an **activity**

A configuration describes the active entities in the dimension overlaid by the service layer

The nodes of a configuration represent entities that execute some computation

The nodes of a configuration may derive form different initial modules

The nodes of a configuration correspond to the nodes of a SRML module

The computation of a node in a configuration is modelled by the specification of the node in the SRML module
During the execution of a module instance,

- some user may launch a new activity from the top layer
- some event happening in an existing node may trigger the service discovery

In these cases we have a **dynamic reconfiguration**

A SRML module specifies configuration policies to model a number of aspects of the dynamic reconfiguration:

- when the reconfiguration should happen (for both Activities and Services)
- how a new instance should be initialized (for both Activities and Services)
- which functional properties the Activity/Service requires to the discovered services
- which non-functional properties the Activity/Service requires to the discovered services

**Configuration Policies**

- Internal configuration policies
- Business protocols
- Internal configuration policies

SRML do not describe the discovery process itself that we assume provided by the middleware
Dynamic Reconfiguration

- The trigger (e.g., intMO) becomes true
- Discovery (in a repository, through a broker, etc.)
- Matchmaking (of Business Protocols giving syntactic and behavioural description)
- Ranking (External policies expressing SLA constraints)
- Selection
- Binding (Reconfiguration)
Dynamic Reconfiguration

... and binding
Internal Configuration Policies

- **Internal configuration policies**: concern aspects related with the instantiation of the module or those reconfiguration issues that do not involve negotiation with external parties

  - the initialization of service components (when declared in the module)
    
    **BA**: BookingAgent
    
    \[
    \text{intBA} \quad \text{init}: \ S=\text{START} \land \text{logged}=\text{false} \\
    \text{intBA} \quad \text{term}: \ S=\text{END\_UNBOOKED} \lor \\
    \quad \quad (S=\text{END\_PAID} \land \text{today}>=\text{bookTrip.out}) \lor \\
    \quad \quad S=\text{END\_COMPENSATED} \\
    \]

  - the triggering of the discovery of required services

    **FA**: FlightAgent
    
    \[
    \text{trigger}: \ BA.\text{bookFlight} \quad \text{?} \quad \text{(or default)} \\
    \]

    **PA**: PayAgent
    
    \[
    \text{trigger}: \ BA.\text{bookFlight} \quad \text{✔} \quad \text{?} \\
    \]

    **HA**: HotelAgent
    
    \[
    \text{trigger}: \ BA.\text{bookFlight} \quad \text{⊗} \quad \text{?} \land BA.\text{bookFlight.\text{Reply}} \\
    \]
The complete example:

- see notes page 50...

```plaintext
MODULE TravelBooking is

DATATYPES

sorts: username, password, usrdatal, bool, fcode,
hcode, pcode, airport, date, paydata, accountn,
moneyvalue, serviceId, nat

COMPONENTS

BA: BookingAgent

intBAinit: s=START \land logged=false
intBAterm: s=END_UNBOOKED
\lor (s=END_PAYED \land today>bookTrip.out)
\lor s=END_COMPENSATED
```
The complete example:

see notes page 50...

<table>
<thead>
<tr>
<th>PROVIDES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR:</strong> Customer</td>
</tr>
<tr>
<td><strong>CR</strong> Customer</td>
</tr>
<tr>
<td>r&amp;s getProposal</td>
</tr>
<tr>
<td>idData</td>
</tr>
<tr>
<td>income</td>
</tr>
<tr>
<td>preferences</td>
</tr>
<tr>
<td>✗ proposal</td>
</tr>
<tr>
<td>cost</td>
</tr>
<tr>
<td>r&amp;s login</td>
</tr>
<tr>
<td>✗&gt;usr</td>
</tr>
<tr>
<td>pwd</td>
</tr>
<tr>
<td>snd ackRefundSnd</td>
</tr>
<tr>
<td>✗&gt; amount</td>
</tr>
</tbody>
</table>
The complete example:

see notes page 50...

REQUIRES

FA: FlightAgent
    intFA\(\text{trigger}\): BA.bookTrip\(\oplus\)?

PA: PayAgent
    intPA\(\text{trigger}\): BA.bookTrip\(\checkmark\)?

HA: HotelAgent
    intHA\(\text{trigger}\): BA.bookFlight\(\otimes\)
    \(\land\) BA.bookFlight.Reply

USES

DB: UserDB

EXTERNAL POLICY

***

WIRES

***

END MODULE
Just two notes about wires...

see notes page 50...

**REQUIRES**

<table>
<thead>
<tr>
<th>FA:</th>
<th>FlightAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>intFAtrigger: BA.bookTrip?</td>
<td></td>
</tr>
</tbody>
</table>

---

**WIRES**

<table>
<thead>
<tr>
<th>BA BookingAgent</th>
<th>$c_i$</th>
<th>$BF$</th>
<th>$d_i$</th>
<th>FA FlightAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>s &amp; r bookFlight from to out in traveller</td>
<td>$S_i$</td>
<td>$R_i$</td>
<td>$r &amp; s$ lockFlight from to out in traveller</td>
<td>$S_i$, $R_i$</td>
</tr>
<tr>
<td>amount beneficiary payService</td>
<td>$O_i$</td>
<td>$O_i$</td>
<td>$O_i$, $O_i$</td>
<td>$O_i$, $O_i$</td>
</tr>
<tr>
<td>send payAck proof status</td>
<td>$S_i$, $i_i$</td>
<td>$R_i$, $i_i$</td>
<td>rcv payAck proof status</td>
<td></td>
</tr>
<tr>
<td>rev ackRefundRcv amount</td>
<td>$R_i$</td>
<td>$S_i$, $i_i$</td>
<td>$snd$ payRefund amount</td>
<td></td>
</tr>
</tbody>
</table>

---
Just two notes about wires...

See notes page 50...

<table>
<thead>
<tr>
<th>PROVIDES</th>
<th>WIRES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR</strong> Customer</td>
<td><strong>BA</strong> BookingAgent</td>
</tr>
<tr>
<td>r&amp;s getProposal</td>
<td>r&amp;s getProposal</td>
</tr>
<tr>
<td>idData</td>
<td>idData</td>
</tr>
<tr>
<td>income</td>
<td>income</td>
</tr>
<tr>
<td>preferences</td>
<td>preferences</td>
</tr>
<tr>
<td>Xproposal</td>
<td>Xproposal</td>
</tr>
<tr>
<td>cost</td>
<td>cost</td>
</tr>
<tr>
<td>r&amp;s login</td>
<td>r&amp;s login</td>
</tr>
<tr>
<td>usr</td>
<td>usr</td>
</tr>
<tr>
<td>pwd</td>
<td>pwd</td>
</tr>
<tr>
<td>snd ackRefundSnd</td>
<td>snd ackRefundSnd</td>
</tr>
<tr>
<td>amount</td>
<td>amount</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>c_1</th>
<th>CB</th>
<th>d_1</th>
<th>BA BookingAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_1</td>
<td>I(airport, airport, date, date)</td>
<td>R_1</td>
<td>r&amp;s bookTrip from to</td>
</tr>
<tr>
<td>i_1</td>
<td>O(fcode, hcode, moneyvalue)</td>
<td>i_1</td>
<td>out in fconf hconf</td>
</tr>
<tr>
<td>i_2</td>
<td></td>
<td>i_1</td>
<td>amount</td>
</tr>
</tbody>
</table>

| S_1 | I(username, password) | R_1 | r&s login usr pwd |
| i_1 | | i_1 | |

| R_1 | I(moneyvalue) | S_1 | snd ackRefundSnd amount |
| i_1 | | i_1 | |
Agenda

- Internal configuration policies
- External configuration policies
- Non-functional properties and SLA
SLA in SRML

- SRML supports service selection based on QoS
- Model for non-functional requirements of a dynamically changing configuration
- QoS in SRML relies on
  - c-semirings to model satisfiability
  - CSP to model the dynamic reconfiguration of constraints concerning QoS

Semiring-based constraint satisfaction and optimization.
Journal of the ACM (JACM) 44(2): 201-236
External policies in SRML

EXTERNAL POLICY

\[
<[0..1], \text{max, min, 0, 1}>
\]

SLA VARIABLES

\{HA.DIST2CENTRE, HA.DIST2METRO\}

\{FA.BOOKFEE, CR.BOOKFEE\}

\{CR.PERC, FA.PERC\}

CONSTRAINTS

\(C_1: \{HA.DIST2CENTRE, HA.DIST2METRO\}\)

\[def_1(d, p) = \begin{cases} 1 & \text{if } d < 1000 \text{ or } p < 100 \\ 200/p & \text{otherwise} \end{cases}\]

\(C_2: \{CR.BOOKFEE\}\)

\[def_2(n) = \begin{cases} 1 & \text{if } n > 5 \\ 0 & \text{otherwise} \end{cases}\]

\(C_3: \{CR.BOOKFEE, FA.BOOKFEE\}\)

\[def_3(d, p) = \begin{cases} 1 - 1/(d-p+1) & \text{if } d > p \\ 0 & \text{otherwise} \end{cases}\]

\(C_3: \{CR.PERC, FA.PERC\}\)

\[def_3(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}\]
A c-semiring is an algebraic structure $\langle A, +, \times, 0, 1 \rangle$ where:

- $A$ is a set of values such that $\{0, 1\} \subseteq A$
- $+$ is a binary operation on $A$ that is commutative, associative, idempotent and with unit element 0
- $\times$ is another binary operation on $A$ that is commutative, associative with unit element 1 and absorbing element 0
- $\times$ distributes over $+$

$A$ is the domain of the degree of satisfaction
- $\{0, 1\}$ for yes/no
- $[0, 1]$ for intermediate degrees
A c-semiring is an algebraic structure \(<A,+,\times,0,1>\) where:

1. \(A\) is a set of values such that \(\{0,1\} \subseteq A\)
2. \(+\) is a binary operation on \(A\) that is commutative, associative, idempotent and with unit element 0
3. \(\times\) is another binary operation on \(A\) that is commutative, associative with unit element 1 and absorbing element 0
4. \(\times\) distributes over \(+\)

\(+\) is a comparison primitive

\(a < b \iff a + b = b\) (\(b\) is better than \(a\))

\(<\{0,1\},\lor,\land,0,1>\)
\(<[0,1],\max,\min,0,1>\)
What is a c-semiring?

A c-semiring is an algebraic structure $\langle A,+,\times,0,1\rangle$ where:

- $A$ is a set of values such that $\{0,1\} \subseteq A$
- $+$ is a binary operation on $A$ that is commutative, associative, idempotent and with unit element 0
- $\times$ is another binary operation on $A$ that is commutative, associative with unit element 1 and absorbing element 0
- $\times$ distributes over $+$

$\times$ is a composition primitive

$\langle [0,1],\lor,\land,0,1\rangle$

$\langle \{0,1\},\max,\min,0,1\rangle$
SLA in SRML

A constraint system is a triple < S, D, V > where

- S is a C-semiring
- D is a finite set (domain of possible elements taken by the variables)
- V is a totally ordered set (of variables)

A constraint is a pair < def, con > where

- con ⊆ V is called the type of the constraint
- def : D_{|con|} → A is the value (mapping) of the constraint

deg <a_1, a_2, ..., a_{|con|}>

degree of satisfaction
SLA in SRML

- def : D_{\text{con}} \rightarrow A \quad \text{is the value (mapping) of the constraint}

\[<a_1, a_2, \ldots, a_{|\text{con}|}>\]

\text{degree of satisfaction}

- For example, if we have \( V = \{ \text{cost, waitingTime} \} \) then def could map:

\[<50,1> \rightarrow 1 \text{ (if I pay 50 and I wait 1 day I am happy)}\]
\[<50,2> \rightarrow 0.5 \text{ (If I wait 2 days buy then I pay only 50 I am happy)}\]
\[<100,1> \rightarrow 0.5 \text{ (if I pay 100 and I wait 1 day I am half happy)}\]
\[<0,7> \rightarrow 0 \text{ (if I have to wait more than 7 days I am unhappy, even if it is free)}\]

- Defining \text{def} through enumeration could be not practical as there may be infinite values for the variables to consider

- We use functions
External policies in SRML

EXTERNAL POLICY

\([0..1], \text{max, min, 0, 1}\)

SLA VARIABLES

FA.BOOKFEE, CR.BOOKFEE
CR.PERC, FA.PERC

CONSTRAINTS

\{HA.DIST2CENTRE, HA.DIST2METRO\}

\[def_1(d, p) = \text{if } d < 1000 \text{ or } p < 100 \text{ then 1 otherwise } 200/p\]

If the hotel is less than one Km from the centre or less than 100 m from the metro station then the degree of satisfaction is 1 (maximal).

Otherwise the degree of satisfaction is inversely proportional to the distance from the metro station.
External policies in SRML

**EXTERNAL POLICY**

\(<[0..1], \text{max, min, 0, 1}>\)

**SLA VARIABLES**

- HA.DIST2CENTRE, HA.DIST2METRO
- FA.BOOKFEE, CR.BOOKFEE
- CR.PERC, FA.PERC

**CONSTRAINTS**

\[ C_1: \{\text{HA.DIST2CENTRE, HA.DIST2METRO}\} \]

\[ def_1(d, p) = \text{if } d<1000 \text{ or } p<100 \text{ then } 1 \text{ otherwise } 200/p \]

\[ \{\text{CR.BOOKFEE}\} \]

\[ def_2(n) = \text{if } n>5 \text{ then } 1 \text{ otherwise } 0 \]

The booking fee that the customer will agree to pay to TravelBooking must be greater than 5£
External policies in SRML

EXTERNAL POLICY

\(<[0..1], \max, \min, 0, 1>\)

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO

CR.PERC, FA.PERC

CONSTRAINTS

\[ C_2: \{CR.BOOKFEE\} \]

\[ def_2(n) = \text{if } n > 5 \text{ then } 1 \text{ otherwise } 0 \]

\[ \{CR.BOOKFEE, FA.BOOKFEE\} \]

\[ def_3(d,p) = \text{if } d > p \text{ the } 1 - 1/(d-p+1) \text{ otherwise } 0 \]

The booking fee of asked by the flight agent must be lower than the booking fee asked by TravelBooking to the customer.

Specifically, the degree of satisfaction of TravelBooking is directly proportional to the difference of the fee gained (from Customer) and the fee paid (fo FlightAgent).
External policies in SRML

EXTERNAL POLICY

$<[0..1], \text{max, min, 0, 1}>$

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO
FA.BOOKFEE, CR.BOOKFEE

CONSTRAINTS

... 

$C_3: \{\text{CR.BOOKFEE, FA.BOOKFEE}\}$

$def_3(d, p) = \text{if } d > p \text{ the } 1 - \frac{1}{d-p+1} \text{ otherwise 0}$

$\{\text{CR.PERC, FA.PERC}\}$

$def_3(a, b) = \text{if } a = b \text{ then 1 otherwise 0}$

The percentage of refund promised to the customer must be the same as the one offered by the flight agent.
Reconfiguration and SLA

\[ P_1 = \langle C_1, \text{con}_1 \rangle \]

\[ P_2 = \langle C_2, \text{con}_2 \rangle \]

- Compatibility checked by combining
- the projection of \( P_1 \) on the attributes of HA
- the projection of \( P_2 \) on the attributes of CS
The complete example:

see notes page 50...

BUSINESS PROTOCOL FlightAgent is

INTERACTIONS
r&s lockFlight
    from, to: airport,
    out, in: date,
    traveller: userdata
    fconf: fcode
    amount: moneyValue,
    beneficiary: accountn,
    payService: serviceId
rcv payAck
    proof: pcode
    status: bool
snd payRefund
    amount: moneyValue

BEHAVIOUR
initiallyEnabled lockFlight?
    (lockFlight?? ∧ lockFlight.Reply) enables payAck?
    (payAck?? ∧ payAck.status) enables lockFlight??
    until today=lockFlight.out
    (lockFlight?? ∧ today<lockFlight.out)
    ensures payRefund!!

How should we change (restrict) the behaviour of FlightAgent to allow the compensation of lockFlight to be accepted only until 5 days before the trip?

And to ensure the refund only if the compensation occurs more than 5 days before the trip?
The complete example:

- see notes page 50...

```plaintext

BUSINESS PROTOCOL FlightAgent is

INTERACTIONS

r&s lockFlight
  参数:from, to: airport,
  参数:out, in: date,
  参数:traveller: userdata
  参数:fconf: code
  参数:amount: moneyValue,
  参数:beneficiary: accountName,
  参数:payService: serviceId

rcv payAck
  参数:proof: code
  参数:status: boolean

snd payRefund
  参数:amount: moneyValue

BEHAVIOUR

initiallyEnabled lockFlight?;
(lockFlight?? & lockFlight.Reply) enables payAck?;
(payAck?? & payAck.status) enables lockFlight??;
until today+5< lockFlight.out
(lockFlight?? & today+5< lockFlight.out)
ensures payRefund??;
```

- Remember that we were negotiating the parameter FA.KD in the SLA?
- And to ensure the refund only if the compensation occurs more than 5 days before the trip?
Can BookTrip provide the business protocol Customer, relying on FlightAgent?

- Customer is allowed to compensate the flight, e.g., the day before the trip but the flight agent does not allow this and will not provide any refund.

- In theory BookTrip could provide Customer BUT it should implement the orchestration accordingly (and pay the refund by itself if the customer compensates, e.g., the day before the trip!)
Can BookTrip provide the business protocol Customer, relying on FlightAgent?

Yes, because we provide a more restrictive condition to customer than the one we can rely on.

On the down side, we should not restrict Customer more than what if necessary otherwise the customer may choose another service that provides better conditions.

The properties (functional and non functional) provided and required should be well tuned.
External policies in SRML

EXTERNAL POLICY

\<[0..1], \text{max, min, 0, 1}\>

SLA VARIABLES

CR.KD, FA.KD,

...

CONSTRAINTS

C_3: \{CR.KD, FA.KD\}

\[def_3(a, b) = 1 \text{ if } a = c \text{ and } 0 \text{ otherwise}\]

The values of KD provided to the customer has to be the same as the one agreed with the flight agent.
We can use the SLA variables in the business protocols.

- The value of CR.KD is defined when the instance of BookTrip is created (and bound to the service/activity of the customer)
- The value of FA.KD is defined when a flight service is discovered, selected and instantiated as soon as BookTrip is instantiated (if the trigger of FA is defined as “true”)
- at the first attempt of interaction of BookTrip with the flight agent (if the trigger is “default”)
- whenever the trigger of FA becomes true...
The business role of BookingAgent, that orchestrate the interactions between the customer and the flight agent has to be “tuned” with the business protocols.

Also business role can depend on SLA variables.
External policies in SRML

**EXTERNAL POLICY**

\([0..1], \text{max, min, 0, 1}\)

**SLA VARIABLES**

PA.ServiceID

**CONSTRAINTS**

\[ C_2: \{\text{PA.ServiceId}\} \]

\[ \text{def}_2(n) = 1 \text{ if } n = \text{FA.lockFlight.payService} \text{ and } 0 \text{ otherwise} \]

ServiceId is the service identifier of PA (similar to an URI)

The value of ServiceId is communicated by FA during the orchestration

BookTrip does not perform an actual discovery but binds to the pay agent specified by the flight agent
Problems

EXTERNAL POLICY
\([-0.1], \text{max}, \text{min}, 0, 1\]

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS

... 

C_2: \{CR.BOOKFEE\} \quad \text{def}_2(n) = \text{if } n>5 \text{ then } 1 \text{ otherwise } 0

C_3: \{CR.BOOKFEE, FA.BOOKFEE\} \quad \text{def}_3(d,p) = \text{if } d>p \text{ then } 1 - \frac{1}{d-p+1} \text{ otherwise } 0

Change the business role BookingAgent to let customer pay CR.BOOKFEE to the payagent (ignore the problem of “distributing” the amount between TravelBooking and FlightAgent)
Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS


C2: {CR.BOOKFEE} \textit{def}2(n)= \text{if } n>5 \text{ then } 1 \text{ otherwise } 0

C3: {CR.BOOKFEE, FA.BOOKFEE} \textit{def}3(d,p)= \text{if } d>p \text{ then } 1-1/(d-p+1) \text{ otherwise } 0

Change the business protocol Customer to ensure that the parameter amount of refund is equal to the amount of the trip minus the booking fee (which is not refunded)
Problems

EXTERNAL POLICY

$<[0..1], \max, \min, 0, 1>$

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS

... 

$C_2$: $\{\text{CR.BOOKFEE}\}$ $\text{def}_2(n) = \begin{cases} 1 & \text{if } n > 5 \\ 0 & \text{otherwise} \end{cases}$

$C_3$: $\{\text{CR.BOOKFEE}, \text{FA.BOOKFEE}\}$ $\text{def}_3(d,p) = \begin{cases} 1 - \frac{1}{d-p+1} & \text{if } d > p \\ 0 & \text{otherwise} \end{cases}$

Which value is negotiated/defined first: CR.BOOKFEE or FA.BOOKFEE?

How to change to internal reconfiguration policies to negotiate them together?

REQUIRES

| FA: FlightAgent | intFA[trigger: default] |
| PA: PayAgent   | intPA[trigger: BA.bookTrip?] |
Problems

EXTERNAL POLICY

\<[0..1],\text{max, min, 0, 1}>\n
SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM

CONSTRAINTS
Problems

EXTERNAL POLICY

\([0..1], \text{max, min, 0, 1}\)

SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM, HA.BOOKINGFEE

CONSTRAINTS

- if the customer has pets it is mandatory that FA and HA accept pets (FA.PETS=true and HA.PETS=true). If the customer does not have pets then the satisfaction is maximal in either case.

- the miles program of the customer must be the same as the miles program of the flight agent

- if the miles program of the hotel is not as the miles program of the satisfaction is inversely proportional to the booking fee (HO.BOOKINGFEE)
Problems

EXTERNAL POLICY

< [0..1], max, min, 0, 1 >

SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM,
HA.BOOKINGFEE

CONSTRAINTS

\[ C_1: \{ \text{CR.PETS, FA.PETS, HA.PETS} \} \]

\[
\text{def}_1(a, b, c) = \begin{cases} 
\text{if } a = \text{true} \text{ then } (\text{if } b = c = \text{true} \text{ then } 1 \text{ otherwise } 0) \\
\text{otherwise } 1 
\end{cases}
\]

\[ C_2: \{ \text{CR.MILESPROGRAM, FA.MILESPROGRAM} \} \]

\[
\text{def}_2(a, b) = \begin{cases} 
\text{if } a = b \text{ then } 1 \\
\text{otherwise } 0 
\end{cases}
\]

\[ C_3: \{ \text{CR.MILESPROGRAM, HA.MILESPROGRAM, HA.BOOKINGFEE} \} \]

\[
\text{def}_2(a, b, c) = \begin{cases} 
\text{if } a = b \text{ then } 1 \\
\text{otherwise } 1/c + 1 
\end{cases}
\]