Decision Management Community Challenge March 2019 Offering Donated Organs for Transplant

Solution using ALEF, Netherlands Tax Administration

Introduction

ALEF is an acronym of Agile Law Execution Factory. It is a tool built with Jetbrains MPS by the Netherlands Tax Administration (NTA) to specify and test rules and to generate decision services from these rules. These rules are specified in a controlled natural language called RegelSpraak[i]. The NTA uses decision services made in this way for millions of tax decisions each year.

Our understanding of the challenge

Given an offer of organ(s) and a response to the offer the system determines the next action for the offer.

Initially, there is no response to the offer, but solely an offer of donated organ(s). In this initial state, the only next action is to pass the offer on to a candidate who needs this type of organ (narrowed down to heart or lungs by the challenge). Next action in this initial state will be: 'Contact candidate with offer of organ'. Of course, this action depends on the existence of candidates for this organ. Because we have treated the challenge as a *complete slice of reality*[ii], there are 3 named candidates for each organ.

Depending on the response of the candidate, next actions for an offer of a single organ can be:

#	Offer	Response	Next action
1	Heart	none	Contact candidate with offer of organ
2	Heart	'accepted'	Send organ in offer to candidate
3	Heart	'declined'	Contact candidate with offer of organ
4	Heart	'declined'	Actions for offers exhausted. Proceed to fall back phase

In #3 the next action is possible only if there are other candidates for the offered organ who have not been approached with an offer of the organ.

In #4 all the candidates for the offered organ have declined. The organ (as part of the block of organs) can be offered to transplant centers instead of individual candidates. Following the clue in the challenge, we have dubbed this the 'fall back phase'. Please see below.

The challenge states that there is a wait state for the offer of one organ when another organ from the same donor is not declined by all candidates in need of this latter organ.
For example: Both a heart and lungs from the same donor are offered for donation. The heart has not yet been declined by all candidates for it but the lungs were declined by all its candidates. In this case, the lungs are 'waiting' for the heart to be offered to all candidates for it.

Although the challenge speaks of 'no action' in this case, 'waiting' was regarded as an action by us.

So, the next actions for the two offers of organs in the challenge can be[iii]:

	Offer, response and action			Offer, response and action		
#	Offer	Response	Next action	Offer	Response	Next action
1	Heart	none	Contact candidate with offer of organ	Lungs	none	Contact candidate with offer of organ
2	Heart	none	Contact candidate with offer of organ	Lungs	'accepted'	Send organ in offer to candidate
3	Heart	none	Contact candidate with offer of organ	Lungs	'declined'	Contact candidate with offer of organ
4	Heart	none	Contact candidate with offer of organ	Lungs	'declined'	Actions for offers exhausted. Proceed to fall back phase
5	Heart	'accepted'	Send organ in offer to candidate	Lungs	'accepted'	Send organ in offer to candidate
6	Heart	'accepted'	Send organ in offer to candidate	Lungs	'declined'	Contact candidate with offer of organ
6	Heart	'accepted'	Send organ in offer to candidate	Lungs	'declined'	Actions for offers exhausted. Proceed to fall back phase
7	Heart	'declined'	Contact candidate with offer of organ	Lungs	'declined'	Contact candidate with offer of organ
8	Heart	'declined'	Contact candidate with offer of organ	Lungs	'declined'	Hold offer while waiting for other offers in block
9	Heart	'declined'	Actions for offers exhausted. Proceed to fall back phase	Lungs	'declined'	Actions for offers exhausted. Proceed to fall back phase

#6 is the scenario where one organ was accepted by a candidate and the other organ will be offered to transplant centers (the fall back phase described below).

#8 is the 'wait action' for the offer of the donor's lungs (as per the example above).

#9 is the scenario where both organs have been declined by all candidates and the block of both heart and lungs will be offered to transplant centers. The action is then 'proceed' to fall back phase.

Fall back phase

In the fall back phase, a block of organs is offered to transplant centers. The challenge is somewhat inconsistent of its treatment of this block, because a partial accept of a block is possible.

We have dealt with this inconsistency by coupling actions to blocks when the block is declined or accepted, but coupling actions to offers within the block when the block is partially accepted.

Also not clear in the case is whether to treat a single unallocated organ from the previous phase as a block. We decided to regard a single organ block as a block. The possible actions for a single organ block are:

#	Response	Next action for block
10	None	Offer block to transplant center
11	‘accept’	Send block to transplant center
12	‘decline’	Offer block to transplant center
13	‘decline’	Block completely declined. Challenge does not state what to do next’

#11 is the case that the block is accepted.

#13 is the unhappy scenario that no transplant center accepted the block. The process ends.

For a multiple organ block, there are several other possibilities. The possible actions for such a block are:

#	Response	Next action for block
14	None	Offer block to transplant center
15	‘accept’	Send block to transplant center
16	‘accept’	Send organ in block to transplant center
17	‘decline’	Offer block to transplant center
18	‘decline’	Block completely declined. Challenge does not state what to do next’
19	‘decline’	'Actions for organ(s) in block exhausted. Challenge does not state what to do next'
20	‘partial accept’	Send organ in block to transplant center Offer remaining organs in block to transplant center
21	‘partial accept’	Send organ in block to transplant center Actions for organ(s) in block exhausted. Challenge does not state what to do next

#14 is the case that one organ in a multiple organ block is accepted, while the other organs were previously accepted.

#18 is the case when all organs in the block have been declined by all transplant centers. The process ends.

#19 is the scenario that the transplant center with the lowest priority declines the block, but a partial accept was received from a another transplant center. The process ends.

#20 A partial accept results in 2 actions for the block: 1 for sending the accepted organ to the transplant center and 1 for offering the remaining organ to other transplant centers.

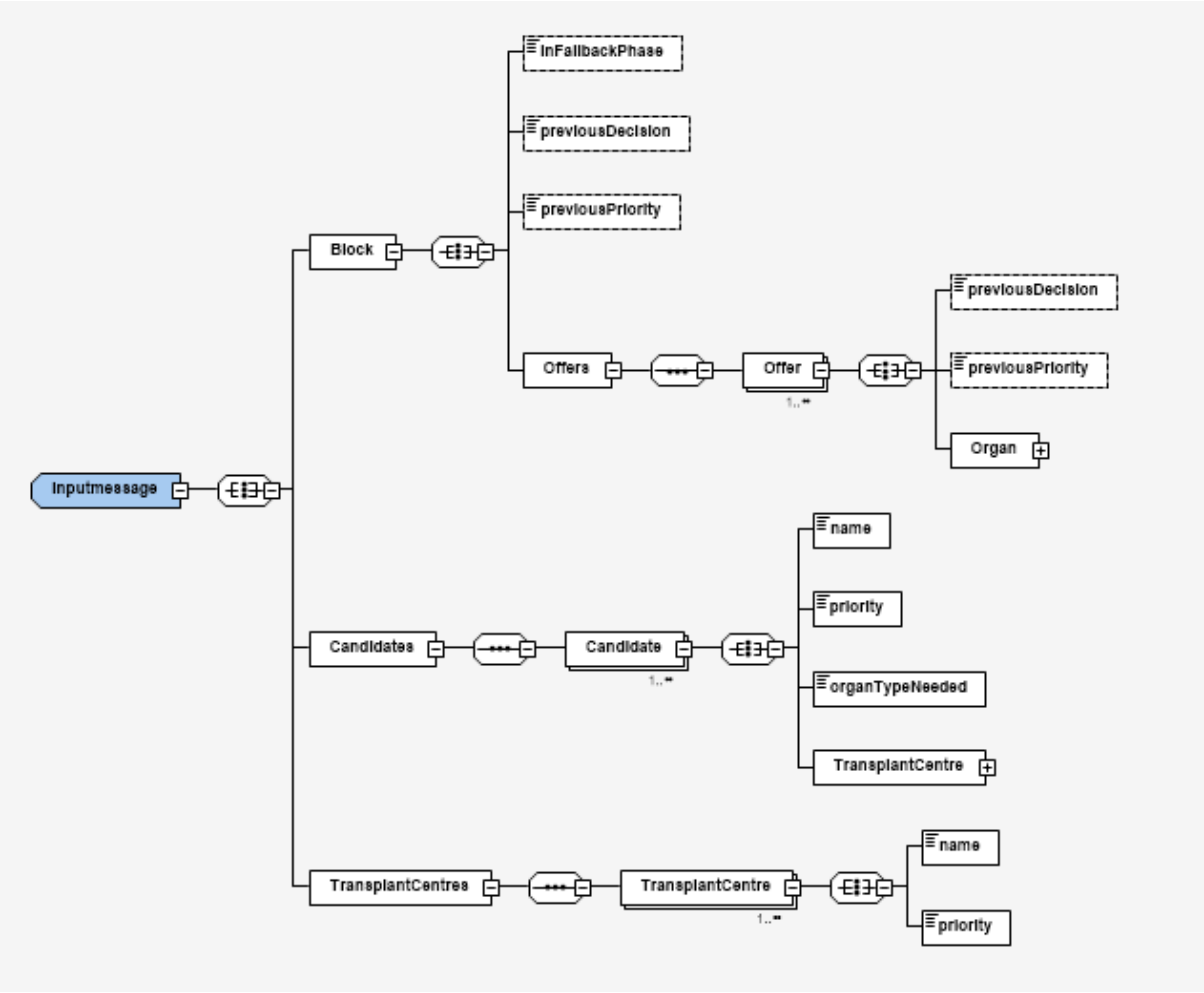
#21 is the situation that the transplant center with the lowest priority accepts one organ in the block, leaving one organ unallocated. The process ends.

The solution in ALEF

ALEF generates stateless services only. This limitation leads us to rely on another client application to keep state and feed the response to offer of organs or blocks of organ(s) to our service.

We explored a solution in which the output of the service could be fed back into the service to cycle through the whole process and to minimize 'state keeping' by the consuming application. This was not feasible at the moment[iv].

So, in the end, we built a service that relies on a consuming application for keeping state. This service has the following request and response[v]:



The external application calls the service with a request containing:

- a Block of offers of one organ each.
- A list of candidates and the transplant center that treats them
- A list of transplant centers

The block has a status variable 'InFallBackPhase' that should be set to true when the process is in the fall back phase. Setting this status variable happens as the result of an Action (see #6 and #9 above for example). This variable has a default value of 'false', so it can be left out of the request initially. When a block is in fallback phase, the elements previousDecision and previousPriority must be set.

These elements are also present with individual offers, where they are used when not in the fall back phase.

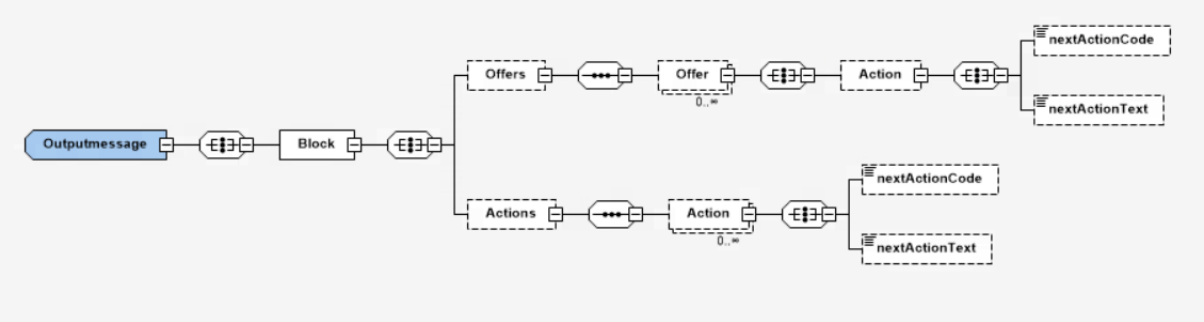
Each Offer has two variables to communicate the latest response of candidates to the service: previousDecision and previousPriority. previousDecision is an enumeration with values ‘accepted’, ‘declined’ and ‘partiallyAccepted’. The last value is solely used for blocks in the fall back phase.

previousPriority is the priority of the candidate responding to an offer. It has a default value of ‘0’, so it can be left out in the initial request.

Together, previousDecision and previousPriority, provide a shorthand notation of the decision in response to an offer and who made this decision.

The list of candidates is straightforward: There is one list of alle candidates, the type of organ they need[vi] and the transplant centre that treats the candidate.

The list of transplant centers in the inputMessage is used in the fallback Phase. It should be optional in the first phase, but that would take additional input validation logic, so we decided to make this list a mandatory element.



The outputmessage of the service has an Action element that is either linked to an offer or to a block (the former when not in the fall back phase, the latter when in the fall back phase).

The nextActionCode is an enumeration with values corresponding to the actions above. Please see Part 2 for the entire schema.

Logic of the service

Working our way back from a result of the service, we see a typical rule for an ‘offer to eligible candidate’. This rule applies when not in the fall back phase and given that there are candidates to whom an offer of an organ can be made.

```
Rule next action for offer 01
  validity always
    An offer to eligible candidate has action for offer with next action code equal to 'Contact candidate with offer of organ' en
    next action text equal to "Contact «the name of eligible candidate to contact with offer of the offer to eligible candidate» at
        «the name of transplant centre of candidate of eligible candidate to contact with offer of the offer to eligible candidate» with offer of
        «the type of organ in offer of the offer to eligible candidate»"
    if block with offers of the offer to eligible candidate is not in fallback phase.
```

The eligible candidate is the specific candidate the offer will be made to. For this relationship, a specific fact type is used:

```
Binairy facttype  Offer to eligible candidate
    the offer to eligible candidate              Offer
    the eligible candidate to contact with offer Candidate
one  offer to eligible candidate  is offered to  one  eligible candidate to contact with offer
```

To instantiate this fact type, we use a fact creation rule:

```
Rule  fact creation eligible candidate to contact with offer 01
  validity   always
     eligible candidate to contact with offer  van  an  Offer  is
     candidate within message  of   message with block  of   block with offers  of  the  Offer
  if  all  of  following  conditions  are  met:
     • the  candidate within message  is  a(n)  eligible candidate
     • the  type  of   organ in offer  of  the  Offer  is  equal  to the  organ type needed  of  the  candidate within message.
```

The first condition points to a characteristic of the candidate: Being an eligible candidate. This characteristic is derived by the rule below:

```
Rule  eligible candidate 01
  validity   always
  A  Candidate  is  an  eligible candidate
  if  exactly  one  of the  offers in block  of   block within message  of his  message with candidate  meets  all  of  following  conditions:
     • the  offer in block  meets  exactly  one  of the  following  conditions:
        •• the  previous decision  of the  offer in block  is  equal  to  'declined'
        •• the  previous decision  of the  offer in block  be  empty
     • the  type  of   organ in offer  of the  offer in block  is  equal to his  organ type needed
     • his  priority  is  equal to the  previous priority  of the  offer in block  plus
       1.
```

The condition is verbose, because we need to relate a candidate to a specific offer of an organ. This is a good time some more details of the object model:

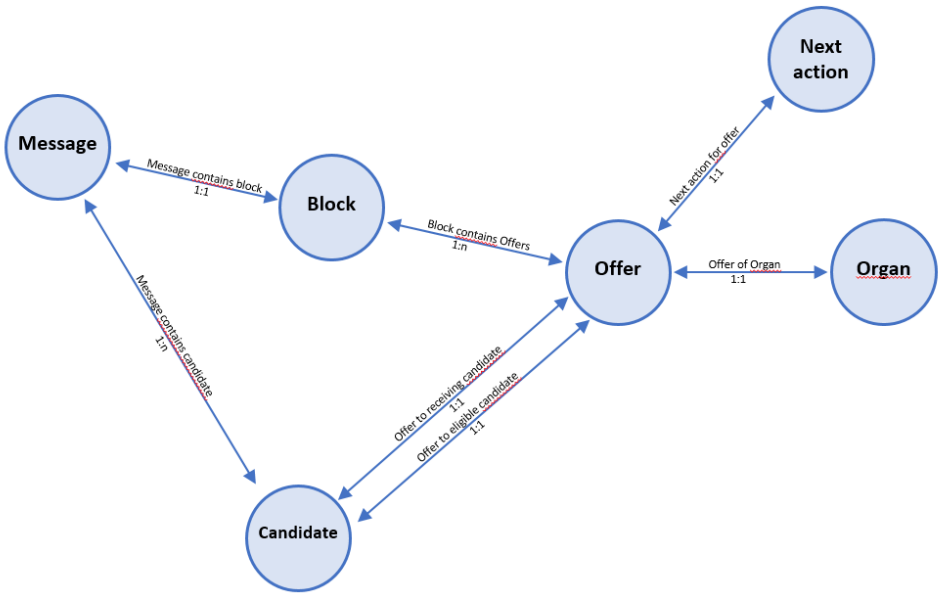


Figure 1 Object model with fact type names for eligible and receiving candidate

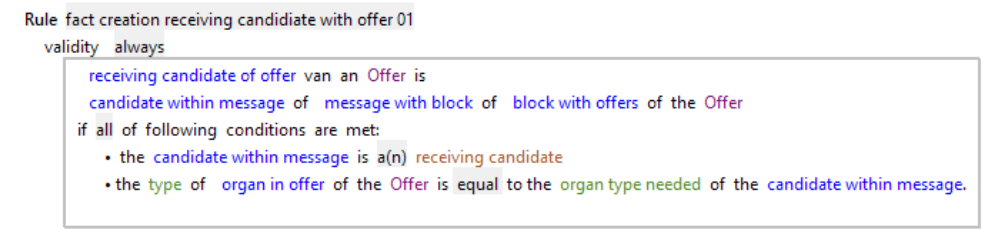
To match a candidate with an organ, we must:

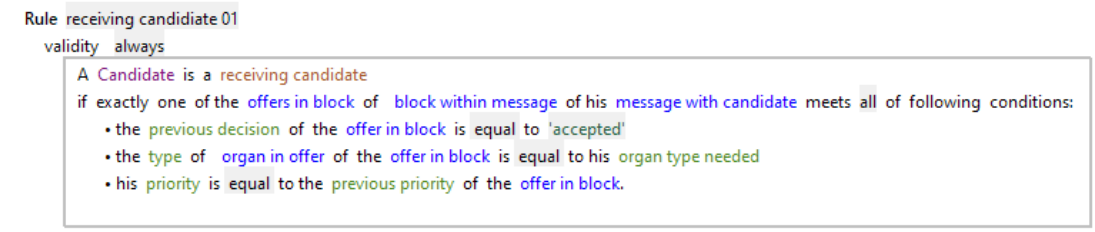
1. Make sure that the service is in initial state (previous decision of offer is empty) or that an earlier offer as been declined
2. compare the type of organ needed by the candidate with the type of organ that belongs to this specific offer
3. check that the priority of the candidate (‘his priority’ in the condition) is 1 higher than the priority of the candidate of the previous decision (see remarks about shorthand above)

All the checks above are done by navigating the fact types in the object model. ALEF turn these into phrases consisting of the role in the fact types, so a fact type navigated in one direction will be phrased differently than the same fact type in the other direction. This can seem verbose, role names can be shortened later to make for easier reading.

The eligible candidate is the person whose name appears in the action text above in rule next action for offer 01. For example: ‘Contact Adam at TC North with offer of heart’. The action text is our way of showing that we have all the information we need to communicate specific instructions for each offer to the consuming application.

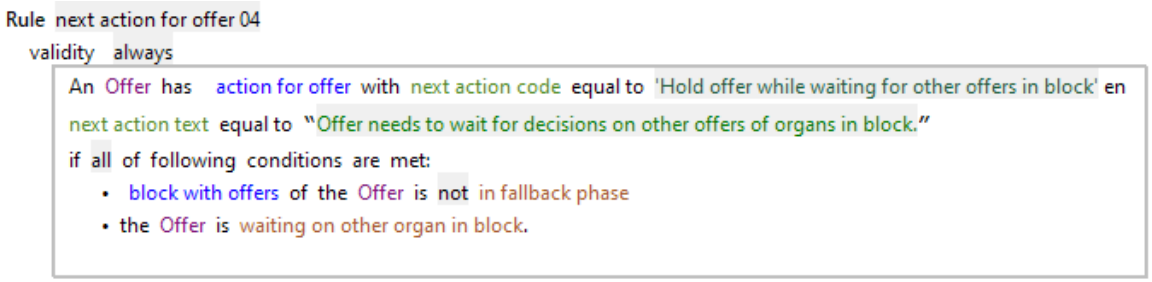
The same principles have been used to derive the ‘receiving candidate’, i.e. the person to whom the organ of an offer must be sent for transplant.





A rather peculiar situation is described in the challenge, where one organ in the block needs to ‘wait’ for another organ to be completely declined by all candidates or be accepted by one candidate for it.

We have solved this by deriving a characteristic of the offer called ‘waiting on other organ in block’



```
Rule offer waiting on other organ in block 01
  validity always
    An Offer is waiting on other organ in block
    if the Offer meets all of following conditions:
      • the number of offers in block of block with offers of the Offer is greater than 1
      • the Offer is completely declined
      • the number of offers in block status accepted of the block with offers plus
        the number of offers in block status completely declined of the block with offers
          is less than the number of offers in block of the block with offers.
```

We would have preferred to specify the last condition as ‘there is **another** offer that meets all of the following conditions: the offer is not completely declined and the offer is not accepted’, but at this time, ALEF does not have the notion of ‘self’ or ‘other’. So we resorted to counting the offers in the block.

An offer is completely declined when:

```
Rule offer completely declined 01
  validity always
    An Offer is completely declined
    if the Offer meets all of following conditions:
      • the previous decision of the Offer is equal to 'declined'
      • the previous priority of the Offer is equal to het aantal
          candidates within message of message with block of block with offers of the Offer die meets the following condition:
        •• his organ type needed is equal to the type of organ in offer of the Offer...
```

Fall back phase

Here, we followed an approach similar to the one above. The main difference being that we now attach actions to blocks instead of offers:

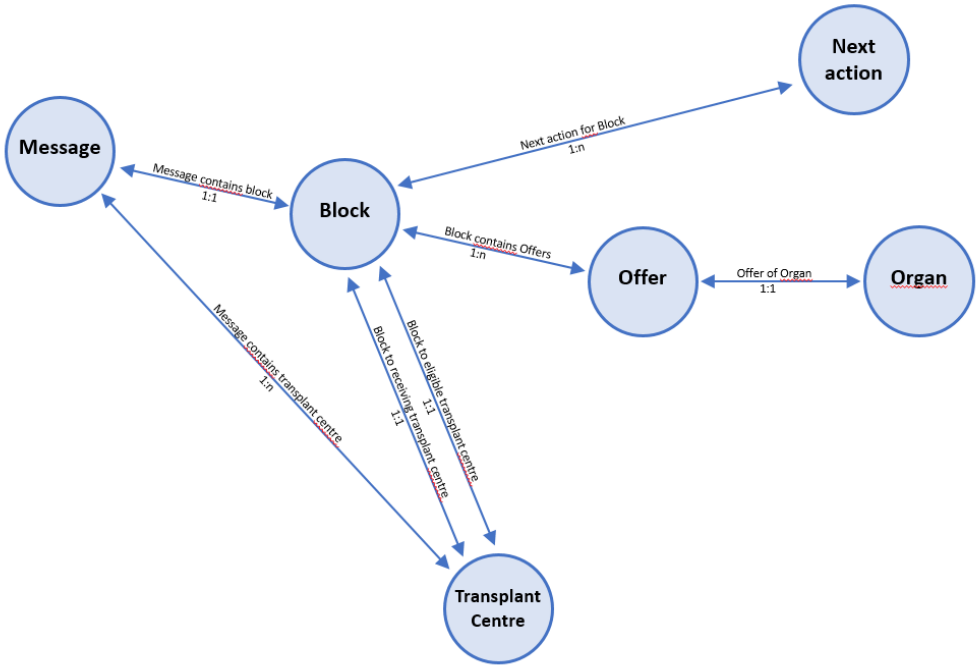


Figure 2 Object model with relevant parts for the fall back phase

Rule next action 01
validity always

A Block has action for block with next action code equal to 'Offer block to transplant center' en
next action text equal to "Contact «the name of eligible transplant centre to contact with block of the Block» with offer of block"
if the Block meets all of following conditions:
- the Block has not one organ completely declined
- the Block has not all organs in block declined
- the Block has not one organ in block accepted
- the Block is not accepted
- the Block is in fallback phase.

In the above rule, the Block as a whole has been declined by a transplant center. This situation corresponds with #17

Rule fact creation eligible transplant centre to contact with block 01
validity always

eligible transplant centre to contact with block van a Block is
transplant centre within message of message with block of the Block
if all of following conditions are met:
- the transplant centre within message is a(n) eligible transplant centre
- the Block is in fallback phase.

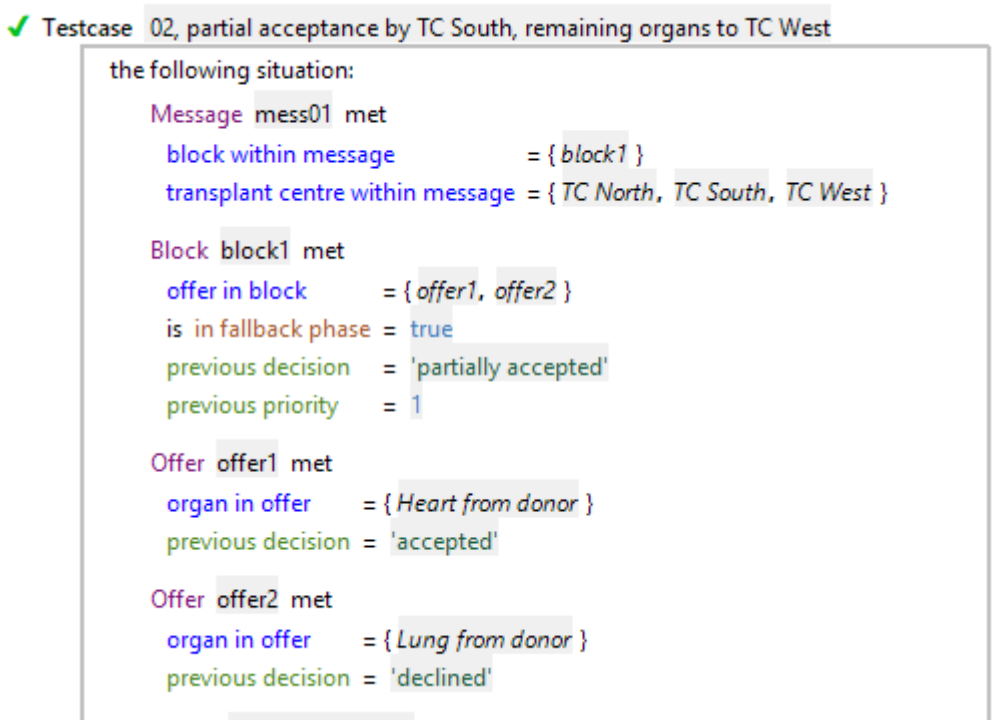
Rule eligible TC 01
validity always

A Transplant centre is an eligible transplant centre
if block within message of message with transplant centre of the Transplant centre meets all of following conditions:
- the block within message meets exactly one of the following conditions:
  - the previous decision of the block within message is equal to 'partially accepted'
  - the previous decision of the block within message is equal to 'declined'
  - the previous decision of the block within message **be** empty
- the priority of the Transplant centre is equal to (the previous priority of the block within message plus 1).

Although organs can be offered as a block to transplant centres, a multiple organ block can be partially accepted. This means we need to make it possible to have different decisions for each offer in the block.

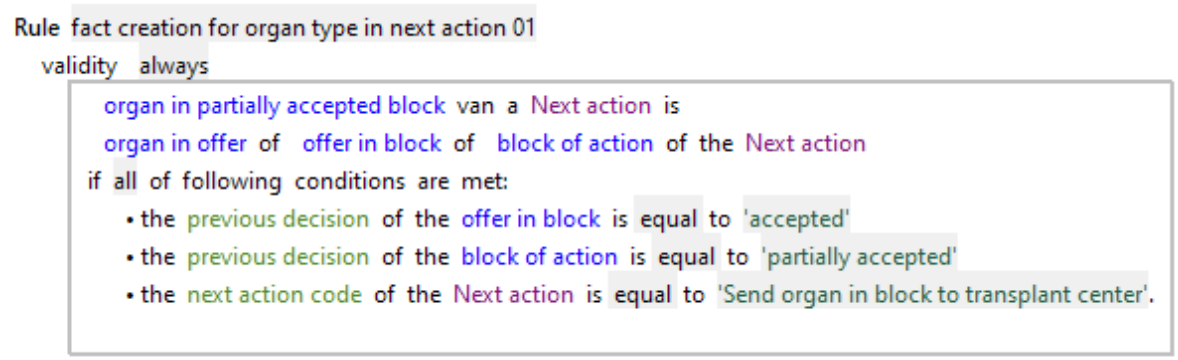
In the case of partial acceptance we have decided to use previous decision attributes for both the block and the offer. This being a special case, for which we cannot validity controls in de xsd of the service, we would normally add validation rules on the input message. To keep things simple, we have not done this here.

Below, part of a test case in ALEF for this situation is shown:

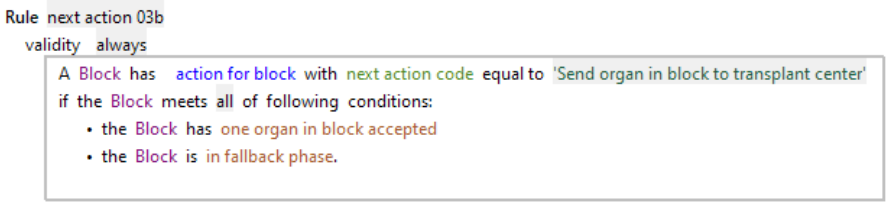


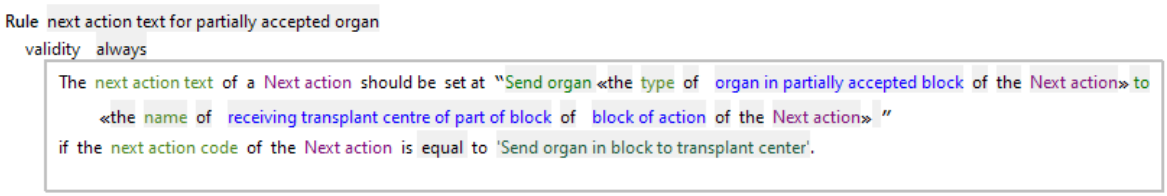
The action text in this case should be: ‘Send organ heart to TC South’.

To point to the specific organ, we use a fact creation rule again:



Using this fact type, we can set the action text for Next Action in two steps:





In all other situations we can set both the action code and the action text in one rule. In this case however, we create the Next Action in rule next action 3b.

It is impossible to also instantiate a fact type (using rule fact creation for organ type in next action 01) to this newly created Object at the same time. This is a timing issue we rarely encounter in practice.

For the full text of the specifications in the xsd schema, the object model, fact types and rules, please see Part 2.

Conclusion

Although the challenge is about a problem unfamiliar to us, ALEF did support our verbal way of modelling very well. We can specify state and object model in a declarative, easily readable way.

ALEF allows us to continuously test our specifications, which we have used to our advantage in building this solution. We can now generate an service, which can be consumed by another application to keep state or we can use a tool like SOAP UI to simulate the entire process.

We have also discovered some improvements for ALEF, which we are happy about.

The LinkedIn challenge team

Our thanks to a developers for providing a translation of RegelSpraak to English[vii].

---

[i] Please see: https://wendbarewetsuitvoering.pleio.nl/ for more (mostly Dutch) information about ALEF and RegelSpraak.

ii As a rule, we adopt a closed world perspective to implement a decision on one tax subject. In this case, we did so to protect ourselves against scope creep, loss of focus and unproductive discussions.

[iii] There are more combinations. We have left out duplicates (regarding A + B as equal to B + A, where A and B are unique combinations of response and next action for the offer of heart and the offer of the lungs).

[iv] In ALEF 2022.3.0, the use of fact types to point to individual instances and strict checks on cycles made this too cumbersome.

[v] Diagrams made with xsd diagram (http://regis.cosnier.free.fr).

vi Limited to one type of organ, corresponding with the challenge

vii Which can be found at: https://github.com/diederikd/RegelspraakEN