Analysis

J. L. De Coi

August 22, 2006

- The following Java-like code tries to catch the logic of the algorithm: the synchronous protocol, as well as other implementation issues entailed by the choice of Java as a language, is not supposed to be necessarily part of the final solution
- Comments are written over the line(s) they refer to
- ```
  Type1[] t1a;
  Type2[] t2a=t1a.method();
  ```

  is a shortcut for

  ```
  Type1[] t1a;
  Type2[] t2a= new Type2[t1a.length];
  for(i=0; i<t1a.length; i++) t2a[i]=t1a[i].method();
  ```

1 Peer

```
Status s; // Initialised to an empty status
Checker c; // It is not included in the negotiation model
TerminationAlgorithm ta;
Filter f;
CredentialSelectionFunction csf;
Policy p;
Peer otherPeer;

void perform(FilteredPolicy fp, Notification[] na){
  Action[] unlocked;
  FilteredPolicy[] fpToSend;

  s.add(fp);
  s.add(na);
  Check[] ca = c.check(na);
```
```

```
  // It is not explicitely included in the negotiation model
  s.add(ca);

  if(ta.terminate(s)){
    send(TERMINATION_MESSAGE, otherPeer);
    return;
  }

  /* Methods extractActions and isLocked are shared among
  each Filter subclass. Indeed they depend only on the
  coding of the Actions, i.e. on the Protune language */
  Action[] toPerform = f.extractActions(fp);

  /* Why do I filter policies even for unlocked actions? */
  FilteredPolicy[] myFp = f.filter(toPerform, p, s);

  for each i
    /* Why should I write f.isLocked(myFp[i]) instead of
    f.isLocked(toPerform[i])? */
    if(f.isLocked(myFp[i])) fpToSend.add(myFp[i]);
    else unlocked.add(toPerform[i]);

  Action[] aa = csf.selectActions(unlocked, s);
  Notification[] naToSend = aa.perform();
  s.add(fpToSend);
  s.add(naToSend);

  Message m = new Message(fpToSend, naToSend);
  send(m, otherPeer);
}
```

Main changes

- Entities `Checker` and `Policy` were added
- `evaluationState` was removed: it is part of the `InferenceEngine` and not of the `Peer`
- Method `terminate` of class `TerminationAlgorithm` has exactly one parameter of type `Status`
- Method `filter` of class `Filter` has exactly three parameters of type `Action`, `Policy` and `Status`
- Method `selectActions` of class `CredentialSelectionFunction` has exactly two parameters of type `Action[]` and `Status`

2 Filter

```
InferenceEngine ie;

filter(Action a, Policy p, Status s){
  Action[] aa = ie.firstStep(a, p, s);
  while(aa.length != 0){
    Notification[] na = aa.perform();
    ie.addToEvaluationState(na);
    aa = ie.secondStep(a, p, s);
  }
  return ie.thirdStep(a, p, s);
  // What about the actions whose evaluation is delayed?
}
```