Algorithm 1 Main Comparison Algorithm

Input CONReq: CONTEXT; CONProvider: ListCON
Output ConMatchedList: ListCON
1: Begin
2: switch choice **do**
3: case ExactCON **do**
$ComparisonResult \leftarrow ModeExactCON(CONReq, CONProvider)$;
4: case PluginCON **do**
$ComparisonResult \leftarrow ModePluginCON(CONReq, CONProvider)$;
5: case SubsumeCON **do**
$ComparisonResult \leftarrow ModeSubsumeCON(CONReq, CONProvider)$;
6: $ConMatchedList \leftarrow null$;
7: for each CONTEXT c of ComparisonResult **do**
//ContextID is a function that returns ID of a certain CONTEXT
8: $ConMatchedList \leftarrow ConMatchedList + ContextID(c)$;
9: End

Algorithm 2 ModeExactCON Algorithm

//This function returns a list of contexts exactly matching a given context
//We used a predefined function NbElement that returns the number of contextual attributes
Variables
Results: CONTEXTS //Matched contexts returned via the function
Compatible: boolean //To define the compatibility between two contexts
1: Begin
2: $Resuls \leftarrow null$;
3: for each CONTEXT c **of** CONTEXTS **do**
4: $i \leftarrow 1$; $Compatible \leftarrow true$;
5: while (i≤ NbElement(c.attribute)) **and** (Compatible=true) **do**
6: $j \leftarrow 1$;
7: while (j≤ NbElement(c.attribute)) **and** (c.attribute(j)!=c.attribute(i)) **do**
8: if (c.attribute(i)!=c.attribute(j)) and (c.value(i)!=c.value(j)) and (j=NbElement(c.attribute)) **then**
9: $Compatible \leftarrow false$;
10: end if
11: $j \leftarrow j + 1$;
12: end while
13: $i \leftarrow i + 1$;
14: end while
15: if Compatible=true **then**
16: $Results \leftarrow Results + c$;
17: end for
18:$ModeExactCON \leftarrow Results$;
19: End

Algorithm 3 ModePluginCON Algorithm

//This function returns a list of contexts that includes a given context
//We used a predefined function NbElement that returns the number of contextual attributes
Variables
Results: CONTEXTS
Compatible: boolean
1: Begin
2: $Resuls \leftarrow null$;
3: for each CONTEXT c **of** CONTEXTS **do**
4: $i \leftarrow 1$; $Compatible \leftarrow true$;
5: while (i≤ NbElement(c.attribute)) **and** (Compatible=true) **do**
6: $j \leftarrow 1$;
7: while (j≤ NbElement(c.attribute)) **and** (!Englobe(c.Name(i),c.Name(j))) **do**
8: if (!Englobe (c.Name(i),c.Name(j)))**and** (!Englobe (c.value(i),c.value(j))) **and** (j=NbElement(c.attribute)) **then**
9: $Compatible \leftarrow false$;
10: end if
11: $j \leftarrow j + 1$;
12: end while
13: $i \leftarrow i + 1$;
14: end while
15: if Compatible=true **then**
16: $Results \leftarrow Results + c$;
17: end for
18:$ModePluginCON \leftarrow Results$;
19: End

Algorithm 4 ModeSubsumeCON Algorithm

//This function returns a list of contexts subsumed by a given context
//We used a predefined function NbElement that returns the number of contextual attributes
//we use the function Englobe
Variables
Results: CONTEXTS
Compatible: boolean
1: Begin
2: $Resuls \leftarrow null$;
3: for each CONTEXT c **of** CONTEXTS **do**
4: $i \leftarrow 1$; $Compatible \leftarrow true$;
5: while (i≤ NbElement(c.attribute)) **and** (Compatible=true) **do**
6: $j \leftarrow 1$;
7: while (j≤ NbElement(c.attribute)) **and** (!Englobe(c.Name(j),c.Name(i))) **do**
8: if (!Englobe (c.Name(j),c.Name(i)))**and** (!Englobe (c.value(j),c.value(i))) **and** (j=NbElement(c.attribute)) **then**
9: $Compatible \leftarrow false$;
10: end if
11: $j \leftarrow j + 1$;
12: end while
13: $i \leftarrow i + 1$;
14: end while
15: if Compatible=true **then**
16: $Results \leftarrow Results + c$;
17: end for
18:$ModeSubsumeCON \leftarrow Results$;
19: End

Algorithm 5 Englobe Algorithm

Variables
CurrentHead: aHead //the head of "A" being reviewed
Ancestors: ListHead//Ancestors of E2
1: Begin
2: $Ancestors \leftarrow null$;
3: if E2=Root(A) **then**
4: $Ancestors \leftarrow null$;
5: else
6: $CurrentHead \leftarrow father(E2)$;
7: $Ancestors \leftarrow father(E2)$;
8: while (CurrentHead != Root(A)) **do**
9: $CurrentHead \leftarrow father(CurrentHead)$;
10: $Ancestors \leftarrow Ancestors + father(CurrentHead)$;
11: end while
12: $Englobe \leftarrow (E1 \in Ancestors)$;
13: End

Algorithm 6 PC-Match Algorithm

```
Input: ReqCON:CONTEXT; ReqCAP:CAPACITY; PCList:List
Output: matchedList:List
Variables: threshold:[0..1]; sim,simCON,simCAP:Real;
1: Begin
2: matchedList ← null;
3: if PCList!=null then
4: for each PC of PCList do
5:       simCON ← CalculSimilarityCON(PC, ReqCON);
6:       simCAP ← CalculSimilarityCAP(PC, ReqCAP);
7:       sim ← (simCON + simCAP)/2;
8:  if sim ≥ threshold then
9:     matchedList ← matchedList + PC;
10:  end for
11:  return matchedList;
11:  End
```

Algorithm 7 CalculSimilarityCON Algorithm

```
Input: ReqCON: CONTEXT; TargetCON: CONTEXT;
Output: degreeSim, simElt:Real;
Variables: PCList:List; simSom,we:Real;
1: Begin
2: ReqElements ← ExtractElementCON(ReqCON);
3: TargetElements ← ExtractElementTarget(TargetCON);
4:  simSom ← 0;
5:  simElt ← 0;
6:  while (! reqElements.isEmpty()) do
7:       EltCurrentReq ← ReqElements.findElementCurrent();
8:      EltCurrentTarget ← TargetElements.findElement(EltCurrentReq);
9:       if EltCurrentTarget.isEmpty() then
10:        simElt ← 0;
11:       else
12:    simElt ← CompareElementCON(EltCurrentCON, EltCurrentTarget);
13:     simSom ← simSom + (simElt * we);
14:     simElt ← simElt + we;
15:  end while
16: return simSom/simElt;
17: End
```

Algorithm 8 CompareElementCON Algorithm

```
Input: ElementReq: String; ElementTarget: String;
Output: score: real;
1: Begin
2: if ElementReq=ElementTarget do
3:  score ← 0;
4:  else
5:  LCSelem ← FindLCS(ElementReq, ElementTarget);
6:  DepthLCS ← Depth(LCSelem); // return the least common subsumer
7:  Depth1 ← Depth(ElementReq); // Depth of ElementReq in the ontology
8:   Depth2 ← Depth(ElementTarget); // Depth of ElementTarget in the
ontology
9: score ← 2 * DepthLCS(Depth1 + Depth2);
10: end if
11: return score
12: End
```

The degree of similarity is calculated using the following formulas:

$$\text{simCON(PC,Request)}=\frac{sim(CON,ReqCON)*wcon}{wcon+wcap}$$

$$\text{simCAP(PC,Request)}=\frac{sim(CAP,ReqCAP)*wcap}{wcon+wcap}$$

$$\mathbf{sim(PC,Request)} = \frac{simCAP + simCON}{2}$$

The degree of similarity is composed of similarity degree of CONTEXT and CAPACITY. We defined weights, represented by CON and CAP to balance these degrees of similarity in order to have the possibility to give more importance to the context in the comparison or vice versa. The value "0" for a weight indicates that the corresponding element is not taken into account.

- **Algorithms of Calculating Similarity:** The formula that calculates the similarity of a context element (either an attribute or a value) is:

$$\mathbf{sim(ReqCON,TargetCON)} = \frac{\sum_{i=1}^{k} sim(ER_i, ET_i) * P_i(w_e)}{\sum_{i=1}^{k} P_i(w_e)}$$

In our work, in order to measure relatedness between two concepts (RequestElement and TargetElement), we applied Wu and Palmer equation [Wu and Palmer, 1994] on our ontology. This equation is mainly based on the depth of the Least Common Subsummer (LCS). According to [Pedersen et al., 2004], The LCS of two concepts A and B is "the most specific concept which is an ancestor of both A and B. Thus, the similarity is twice the depth of the two concepts LCS divided by the product of the depths of the individual concepts as defined in the equation below:

$$\mathbf{sim(ER,ET)} = \frac{2 * Depth(Lcs(ER, ET))}{Depth(ER) + Depth(ET)}$$