Type of derived classes and attributes

This page contains the procedure that X-DEVICE employs for determining the schema of the derived classes and the types of the derived and aggregate attributes.

```
function create_derived_class(Rule) returns Class
   Rule = if Condition then Class(AttExprs)
   SlotDefs, AttLst, EmptyAtt, ElemOrdAtt := ∅
   for_each AttExpr in AttExprs do
      if AttExpr = ^Att:ValueExpr then
         SlotName := Att
         SlotType := string
         SlotReq := mandatory
         SlotCard := single
         SlotDefs := SlotDefs ∪ {slot_desc(SlotName,SlotType,SlotCard,SlotReq)}
         AttLst := AttLst ∪ {SlotName}
      elseif AttExpr = ∅Att then
         SlotName := Att
         SlotType := string
         SlotReq := mandatory
         SlotCard := single
         SlotDefs := SlotDefs ∪ {slot_desc(SlotName,SlotType,SlotCard,SlotReq)}
         EmptyAtt := EmptyAtt ∪ {SlotName}
      elseif AttExpr = !Att:ValueExpr then
         SlotName := Att
         if ValueExpr@SlotType(Patterns) ∈ Condition then
            true
         elseif (I@C(Patterns) ∈ Condition) and (S Op ValueExpr ∈ Patterns) then
            get_slot_desc(slot_desc(S,SlotType,_,_)) => C
         SlotReq := mandatory
         SlotCard := single
         SlotDefs := SlotDefs ∪ {slot_desc(SlotName,SlotType,SlotCard,SlotReq)}
      elseif AttExpr = Att:ValueExpr then
         SlotName := Att
         if ValueExpr@SlotType(Patterns) ∈ Condition then
            true
         elseif (I@C(Patterns) ∈ Condition) and (S Op ValueExpr ∈ Patterns) then
            get_slot_desc(slot_desc(S,SlotType,_,_)) => C
         SlotReq := mandatory
         SlotCard := single
         SlotDefs := SlotDefs ∪ {slot_desc(SlotName,SlotType,SlotCard,SlotReq)}
         ElemOrdAtt := ElemOrdAtt ∪append {SlotName}
   ClassAtts := {elem_ord(ElemOrdAtt), empty(EmptyAtt), att_lst(AttLst)}
   new([Class, ClassAtts ∪ SlotDefs]) => xml_seq

function create_derived_attribute(Rule) returns ∅
   Rule = if Condition then Var@Class(AttExpr)
   if AttExpr = ^Att:ValueExpr then
      SlotName := Att
      SlotType := string
      SlotReq := optional
      if ValueExpr = list(_) or ValueExpr = ord_list(_) then
         SlotCard := list
      else
         SlotCard := single
      put_slot_desc([slot_desc(SlotName,SlotType,SlotCard,SlotReq)]) => Class
      put_att_lst([SlotName]) => Class
   elseif AttExpr = ∅Att then
      SlotName := Att
      SlotType := string
      SlotReq := optional
      SlotCard := single
      put_slot_desc([slot_desc(SlotName,SlotType,SlotCard,SlotReq)]) => Class
      put_empty([SlotName]) => Class
   elseif AttExpr = !Att:ValueExpr then
      SlotName := Att
      if ValueExpr@SlotType(Patterns) ∈ Condition then
         true
      elseif (I@C(Patterns) ∈ Condition) and (S Op ValueExpr ∈ Patterns) then
         get_slot_desc(slot_desc(S,SlotType,_,_)) => C
      SlotReq := optional
      if ValueExpr = list(_) or ValueExpr = ord_list(_) then
         SlotCard := list
      else
```

```
      SlotCard := single
   put_slot_desc([slot_desc(SlotName,SlotType,SlotCard,SlotReq)]) => Class
else
   AttExpr = Att:ValueExpr
   SlotName := Att
   if ValueExpr@SlotType(Patterns) ∈ Condition then
      true
   elseif (I@C(Patterns) ∈ Condition) and (S Op ValueExpr ∈ Patterns) then
      get_slot_desc(slot_desc(S,SlotType,_,_)) => C
   SlotReq := mandatory
   if ValueExpr = list(_) or ValueExpr = ord_list(_) then
      SlotCard := list
   else
      SlotCard := single
   put_slot_desc([slot_desc(SlotName,SlotType,SlotCard,SlotReq)]) => Class
   put_elem_ord([SlotName]) => Class
```