eSML schema (eSourcing Markup Languge)

```
<?xml version="1.0" encoding="uTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Alex Norta (Technische Universiteit Eindhoven) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <!-- PART 1 -->
   <!--  Next, the definition of the data types required in an
   e-contract language  are provided. We use the built in the XML
   schema data types and extend them with the required additional
   attributes for a data item. The common attributes for all data
   items are listed in the group of attributes
   "common_var_attributes". We start with the definition of the
   standard data types and continue with the definition
   of the special data types . At the end of this part of the
   eSML schema, we list three predefined set types (i.e.,
   "state_type", "constraint_type", "currency_type"). These constant
   sets are used as the data type of attributes of contract elements
   that are defined later on in the language schema. -->
   <!-- definition of standard Variable TYPES -->
   <xs:attributeGroup name="common_var_attributes">
      <xs:attribute name="tag_name" type="xs:string"
      use="required"/>
      <xs:attribute name="var_id" type="xs:ID" use="required"/>
      <xs:attribute name="owner" type="xs:IDREFS"
      use="optional"/>
      <xs:attribute name="changeable" type="xs:boolean"
      use="required"/>
      <xs:attribute name="properties" type="xs:IDREFS"
      use="optional"/>
      <xs:attribute name="enabled" type="state_type"
      use="required"/>
      <xs:attribute name="rules_for_change" type="xs:IDREFS"
      use="optional"/>
   </xs:attributeGroup>
   <!-- Tag "name" attribute  is a name from an agreed
   ontology. "var_id" attribute is the unique identifier
   of the data item. "owner" attribute contains a
   reference to an owner of the data item who is allowed
   to update the value of the data item."changeable"
   indicates if the value of the data element can be changed.
   "properties" attribute indicates the properties a data
   element has. "enabled" attribute indicates if the data
   item is currently enabled. The value of this attribute
   is from the defined state_type."rules_for_change" attribute
   is used to indicate the rules in the contract that must
   evaluate to true in order a change to take place. It is
   optional and is used solely for improving the efficiency
   of the monitoring systems. -->
   <xs:complexType name="string_type">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="common_var_attributes"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="real_type">
      <xs:simpleContent>
         <xs:extension base="xs:decimal">
            <xs:attributeGroup ref="common_var_attributes"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="integer_type">
      <xs:simpleContent>
         <xs:extension base="xs:integer">
            <xs:attributeGroup ref="common_var_attributes"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="boolean_type">
      <xs:simpleContent>
         <xs:extension base="xs:boolean">
            <xs:attributeGroup ref="common_var_attributes"/>
         </xs:extension>
```

```
   </xs:simpleContent>
</xs:complexType>
<!--  Sample definition of data items from the List data type.
Additional list types can be defined by designers in the schema
or in the e-contract
itself. -->
<xs:simpleType name="list_of_strings_type">
   <xs:list itemType="xs:string"/>
</xs:simpleType>
<xs:complexType name="list_of_events_type">
   <xs:sequence>
      <xs:element name="event" type="event_type"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<!-- For the definition of data items from the Record data
type, contract designers can use the build in XML Schema
Complex Type element. -->
<!-- Definition of special data types -->
<xs:complexType name="date_type">
   <xs:simpleContent>
      <xs:extension base="xs:date">
         <xs:attributeGroup ref="common_var_attributes"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="time_type">
   <xs:simpleContent>
      <xs:extension base="xs:time">
         <xs:attributeGroup ref="common_var_attributes"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="money_type">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attributeGroup ref="common_var_attributes"/>
         <xs:attribute name="currency" type="currency_type"
         use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="event_type">
   <xs:simpleContent>
      <xs:extension base="xs:boolean">
         <xs:attributeGroup ref="common_var_attributes"/>
         <xs:attribute name="time_of_occurence" type="xs:time"
         use="optional"/>
         <xs:attribute name="date_of_occurence" type="xs:date"
         use="optional"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<!-- For either pushing out or pulling in external data -->
<xs:simpleType name="access_type">
   <xs:restriction base="xs:string">
      <xs:enumeration value="push"/>
      <xs:enumeration value="pull"/>
   </xs:restriction>
</xs:simpleType>
<!-- For referencing variables that are defined in a data_package
-->
<xs:complexType name="external_resource_reference_type">
   <xs:simpleContent>
      <xs:extension base="xs:anyURI">
         <xs:attributeGroup ref="common_var_attributes"/>
         <xs:attribute name="access" type="access_type"
         use="optional"/>
         <xs:attribute name="resource_state"
         type="list_of_resource_types" use="optional"/>
         <xs:attribute name="is_legally_binding"
         type="xs:boolean" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<!--Documents may lso be a part of data packages for data flow.
-->
<xs:complexType name="document_type">
   <xs:sequence>
```

```
        <xs:element name="document_id" type="xs:ID"/>
        <xs:element name="name" type="xs:string"
        maxOccurs="unbounded"/>
        <xs:element name="file_format" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="uri" type="xs:anyURI"
        maxOccurs="unbounded"/>
        <xs:element name="var_section"
        type="variables_def_section" minOccurs="0"/>
        <!-- specify the structure of the document -->
    </xs:sequence>
</xs:complexType>
<xs:complexType name="document_def_section">
    <xs:sequence maxOccurs="unbounded">
        <xs:element name="document_read" type="document_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="document_create" type="document_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="document_update" type="document_type"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="list_of_documents">
    <xs:sequence maxOccurs="unbounded">
        <xs:element name="document" type="document_type"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="internal_resource_reference_type">
    <xs:simpleContent>
        <xs:extension base="xs:anyURI">
            <xs:attributeGroup ref="common_var_attributes"/>
            <xs:attribute name="referenced_package"
            type="xs:IDREFS" use="required"/>
            <xs:attribute name="referenced_variable"
            type="xs:IDREFS" use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<!-- Snippets allow inclusion of predefined contract parts. -->
<xs:complexType name="snippet_type">
    <xs:sequence>
        <xs:any maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="snippet_id" type="xs:ID"
    use="required"/>
</xs:complexType>
<!-- Definition of rule/process state type -->
<xs:simpleType name="state_type">
    <xs:restriction base="xs:string">
        <xs:pattern value="enabled|disabled"/>
    </xs:restriction>
</xs:simpleType>
<!-- Definition of deontic operators -->
<xs:simpleType name="constraint_type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="must"/>
        <xs:enumeration value="may"/>
        <xs:enumeration value="may_not"/>
    </xs:restriction>
</xs:simpleType>
<!-- Definition of static constraints deontic operators -->
<xs:simpleType name="static_constraint_type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="must_observe"/>
        <xs:enumeration value="may_break"/>
    </xs:restriction>
</xs:simpleType>
<!-- Definition of currency types -->
<xs:simpleType name="currency_type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="USD"/>
        <xs:enumeration value="EUR"/>
        <xs:enumeration value="GBP"/>
        <xs:enumeration value="cAD"/>
        <xs:enumeration value="BGN"/>
    </xs:restriction>
</xs:simpleType>
<!-- Definition of resource state types -->
<xs:simpleType name="list_of_resource_types">
```

```
   <xs:list itemType="resource_state_type"/>
</xs:simpleType>
<xs:simpleType name="resource_state_type">
   <xs:restriction base="xs:string">
      <xs:enumeration value="available"/>
      <xs:enumeration value="unavailable"/>
      <xs:enumeration value="changed"/>
   </xs:restriction>
</xs:simpleType>
<!-- PART 2 -->
<!-- In the second part of the eSML schema, we provide
the constructs required for the definition of the integrity,
derivation, and reaction contract rules. The common for
all three rule types attributes are extracted in the
attribute group "common_rule_attributes". A sample set
of operators that can be used in expressions is provided
with an illustrative purpose (e.g., the "unary_operator"
type). The complete set of operators must additionally be
defined. -->
<!-- Definition of common rule attributes. In addition to
the attributes "tag_name", "rule_id", "changeable",
"enabled" explained in the data item definition section,
the following attributes are defined:
The "overrides" attribute indicates which other rule are
overridden by this rule if they fire together. -->
<xs:attributeGroup name="common_rule_attributes">
   <xs:attribute name="tag_name" type="xs:string"
   use="required"/>
   <xs:attribute name="rule_id" type="xs:ID"
   use="required"/>
   <xs:attribute name="enabled" type="state_type"
   use="required"/>
   <xs:attribute name="changeable" type="xs:boolean"
   use="required"/>
   <xs:attribute name="overrides" type="xs:IDREFS"
   use="optional"/>
</xs:attributeGroup>
<!-- Definition of some operators used in expressions
in rules.     -->
<xs:simpleType name="unary_operator">
   <xs:restriction base="xs:string">
      <xs:enumeration value="plus"/>
      <xs:enumeration value="minus"/>
      <xs:enumeration value="sqrt"/>
      <xs:enumeration value="percent"/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name="unary_boolean_operator">
   <xs:restriction base="xs:string">
      <xs:enumeration value="not"/>
   </xs:restriction>
</xs:simpleType>
<!-- Next, we start with the definition of Integrity rules
(distinguishing the two classes of integrity rules, namely
the "state_constraint_rule_type" and the
"dynamic_constraint_rule_type").
The state_constraint_rule_type construct is a simple
Boolean expression. The "assigned_to"
attribute indicates the party (if any) to which the
constraint is assigned. The "type" attribute indicates if
the rule obligates the assigned party to observe the rule
or allows it to break the rule.
The dynamic_constraint_rule_type construct consists of
four elements. If the boolean expression evaluates to
true the rule is in force and the referenced element
"element_ref" (or attribute of the element
"element_attribute_ref") may/may not (depending
on the value of the value of the type attribute) receive
a new value indicating its new state. The
"restricted-nonrestricted_values" ellement contains a set
of allowed/disallowed values.  -->
<!-- Definition of expressions in rule types.
Expressions in rules are presented as strings that
are to be parsed by the contract interpretation software
in order to obtain an expression tree. The expression
strings can be mixed with references to earlier defined
variables. -->
<xs:complexType name="expression" mixed="true">
   <xs:sequence>
```

```
        <xs:element name="expression_variable_ref"
        type="xs:IDREF" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<!-- Definition of the possible rule types and their
sub-constructs -->
<xs:complexType name="state_constraint_rule_type">
    <xs:sequence>
        <xs:element name="rule_conditions"
        type="expression"/>
    </xs:sequence>
    <xs:attributeGroup ref="common_rule_attributes"/>
    <xs:attribute name="assigned_to" type="xs:IDREF"
    use="optional"/>
    <xs:attribute name="type"
    type="static_constraint_type" use="optional"/>
</xs:complexType>
<xs:complexType name="dynamic_constraint_rule_type">
    <xs:sequence>
        <xs:element name="rule_conditions"
        type="expression"/>
        <xs:element name="element_ref" type="xs:IDREF"/>
        <xs:element name="element_attribute_ref"
        type="xs:IDREF" minOccurs="0"/>
        <xs:element name="restricted-nonrestricted_states"
        type="list_of_strings_type"/>
    </xs:sequence>
    <xs:attribute name="type" type="constraint_type"/>
    <xs:attributeGroup ref="common_rule_attributes"/>
</xs:complexType>
<!-- Next, we define the Derivation rule constructs
(distinguishing the two types of derivation rules, namely
"computational_derivation_rule_type" and
"linguistic_derivation_rule_type". The "derived_variable_ref"
attribute contains a reference to the data items the value
of which is derived with this rule. "derivation_expression"
contains a Boolean expression that when evaluating to true
fires the rule.
In "linguistic_derivation_rule_type", the
"property_extended_variable_ref" contains a reference to the
data item that is given a property when the rule fires.
"property_to_be_added_ref" contains a reference to a data
item that contains the property to be added. In this
approach, we require the properties to be defined as data
items in the e-contract, as in this way they can be
referenced by many components and will be from a clearly
defined data type. However, it is possible linguistic
derivation rules to be completely replaced by copier reaction
rules that simply state the new value to be given to the
data item (in a string format). -->
<xs:complexType name="computational_derivation_rule_type">
    <xs:sequence>
        <xs:element name="derived_variable_ref"
        type="xs:IDREFS"/>
        <xs:element name="derivation_expression"
        type="expression"/>
    </xs:sequence>
    <xs:attributeGroup ref="common_rule_attributes"/>
</xs:complexType>
<xs:complexType name="linguistic_derivation_rule_type">
    <xs:sequence>
        <xs:element name="derivation_expression"
        type="expression"/>
        <xs:element name="property_extended_variable_ref"
        type="xs:IDREF"/>
        <xs:element name="property_to_be_added_ref"
        type="xs:IDREF"/>
    </xs:sequence>
    <xs:attributeGroup ref="common_rule_attributes"/>
</xs:complexType>
<!-- Next, the eSML constructs for the support of Reaction
rules follow. A reaction rule can be from one of the three
subtypes, i.e., enabler, executive, and copiers. To support
each of the three types, we provide three sub-constructs
for each of them. Each reaction rule fires when its
"rule_conditions" evaluates to true.
The "enabler_action_type" has two ellements. The
"target_element" contains a reference to the element the
```

```
state of which will be changed. "new_state" indicates the
new state that must be assigned to the element (enabled or
disabled). The "type" attribute indicates the deontic
assignment to a party (defined in the "assigned_to"
attribute), i.e., if a party that is owner of the rule
is obliged/allowed/forbidden to perform the update of the
status of the element. The "executive_action_type"
sub-construct indicates through the "Targets" element the
execution of which process specifications must
be/may be/cannot be started when the rule fires. The "type"
attribute indicates the deontic assignment to a party,
i.e., if a party (defined in the "assigned_to" attribute)
is obliged/allowed/forbidden to start the execution of the
process. The "copier_action_type" sub-construct indicates
which element (the "target_element" element) or its
attribute (the "target_attribute" attribute) is given
a new value (the "new_value" attribute). Again the "type"
attribute indicates the deontic assignment on the owner
of the rule. -->
<xs:complexType name="enabler_action_type">
   <xs:sequence>
      <xs:element name="target_element" type="xs:IDREF"/>
      <xs:element name="new_state" type="state_type"/>
   </xs:sequence>
   <xs:attribute name="type" type="constraint_type"/>
   <xs:attribute name="assigned_to" type="xs:IDREF"
   use="optional"/>
</xs:complexType>
<xs:complexType name="executive_action_type">
   <xs:sequence>
      <xs:element name="targets" type="xs:IDREFS"/>
   </xs:sequence>
   <xs:attribute name="type" type="constraint_type"/>
   <xs:attribute name="assigned_to" type="xs:IDREF"
   use="optional"/>
   <xs:attribute name="repeatable" type="xs:boolean"/>
</xs:complexType>
<xs:complexType name="copier_action_type">
   <xs:sequence>
      <xs:element name="target_element" type="xs:IDREF"/>
      <xs:element name="target_attribute"
      type="xs:string" minOccurs="0"/>
      <xs:element name="new_value"
      type="list_of_strings_type" minOccurs="0"/>
      <!-- besides the assignment of one value, a party
      can be allowed to set one of a set of values (for
      example in the control of a proces status). -->
   </xs:sequence>
   <xs:attribute name="type" type="constraint_type"/>
   <xs:attribute name="assigned_to" type="xs:IDREF"
   use="optional"/>
</xs:complexType>
<xs:complexType name="reaction_rule_type">
   <xs:sequence>
      <xs:element name="rule_conditions" type="expression"/>
      <xs:choice>
         <xs:element name="enabler_action"
         type="enabler_action_type"/>
         <xs:element name="executive_action"
         type="executive_action_type"/>
         <xs:element name="copier_action"
         type="copier_action_type"/>
      </xs:choice>
   </xs:sequence>
   <xs:attributeGroup ref="common_rule_attributes"/>
</xs:complexType>
<!-- Next, we provide the construct for free-text
rules. -->
<xs:complexType name="free_text_rule_type">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attributeGroup
         ref="common_rule_attributes"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<!-- PART 3 -->
<!--  For the definition of process specification constructs
in eSML we use as a foundation XRL (eXchangeable Routing
```

Language). XRL is an instance-based workflow language that uses XML for the representation of process definitions and Petri nets for its semantics. A few modifications have been applied on XRL. These extensions are marked with commentaries. XRL is a prescriptive language which defines explicitly the control flow between activities.
XRL was chosen as process specification language for several reasons. First, its XML representation and short schema definition allow easy integration and extension to suit the eSML goals. Second, the underlying Petri Net semantics in XRL allows the Petri net representation of an XRL process specification to be analyzed using state-of-the-art analysis techniques and tools. Finally, the experience and background of using XRL in the IS group at the Technical University of Eindhoven allowed easier implementation and maintenance in eSML. More information about XRL and its original DTD/schema can be found at:
http://tmitwww.tm.tue.nl/staff/anorta/XRL/xrlHome.html -->

```
<xs:simpleType name="probability">
   <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
   </xs:restriction>
</xs:simpleType>
<xs:attributeGroup name="logistic_attributes">
   <xs:attribute name="cost" type="xs:decimal" u
   se="optional"/>
   <xs:attribute name="min_duration" type="xs:time"
   use="optional"/>
   <xs:attribute name="max_duration" type="xs:time"
   use="optional"/>
   <xs:attribute name="avg_duration" type="xs:time"
   use="optional"/>
   <xs:attribute name="quality" type="probability"
   use="optional"/>
   <!-- quality is defined as how high is the probability
   that the active node (task, block, process) outputs
   the required data-->
</xs:attributeGroup>
<xs:attributeGroup name="contextual_information_attributes">
   <xs:attribute name="goal" type="xs:string"
   use="optional"/>
   <xs:attribute name="risk" type="xs:string"
   use="optional"/>
   <xs:attribute name="project_accomplishment"
   type="xs:string" use="optional"/>
   <xs:attribute name="handbook_reference"
   type="xs:anyURI" use="optional"/>
</xs:attributeGroup>
<xs:group name="common_elements">
   <xs:choice>
      <xs:element name="interface_in"
      type="interface_type"/>
      <xs:element name="interface_out"
      type="interface_type"/>
      <xs:element name="element_id"
      type="xs:ID" minOccurs="0"/>
      <xs:element name="task" type="task_type"/>
      <xs:element name="sequence"
      type="sequence_type"/>
      <xs:element name="any_sequence"
      type="any_sequence_type"/>
      <xs:element name="choice" type="choice_type"/>
      <xs:element name="condition"
      type="condition_type"/>
      <xs:element name="parallel_sync"
      type="parallel_sync_type"/>
      <xs:element name="parallel_no_sync"
      type="parallel_no_sync_type"/>
      <xs:element name="parallel_part_sync"
      type="parallel_part_sync_type"/>
      <xs:element name="parallel_part_sync_cancel"
      type="parallel_part_sync_cancel_type"/>
      <xs:element name="restricted_parallel_sync"
      type="restricted_parallel_sync_type"/>
      <xs:element name="wait_all" type="wait_all_type"/>
      <xs:element name="wait_any" type="wait_any_type"/>
      <xs:element name="while_do" type="while_do_type"/>
      <xs:element name="terminate" type="terminate_type"/>
```

```
        <xs:element name="send_task" type="send_task_type"/>
        <xs:element name="receive_task" type="task_type"/>
        <xs:element name="bi_directional_task"
        type="send_task_type"/>
        <xs:element name="send_transition"
        type="send_transition_type"/>
        <xs:element name="receive_transition"
        type="transition_type"/>
        <xs:element name="bi_directional_transition"
        type="send_transition_type"/>
        <xs:element name="sourcing_sphere"
        type="sourcing_sphere_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="data" type="data_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lock_change" type="lock_change_type"
        minOccurs="0"/>
        <!-- By placing a data definition into the
        common_elements group, data visibility can be defined on
        a group level, e.g., a sequence, parallel_sync, etc. If the
        data definition takes place on a rout level, the range
        stretches over the entire process. -->
        <!-- Adding data definition in the common elements allows
        visibility on a process and block level. -->
    </xs:choice>
</xs:group>
<!-- The lock_type supports workflow data patterns 30 and 31.
Locks that are initially set or not set when a data package is
defined may be altered with the  lock_change tag. -->
<xs:complexType name="lock_change_type">
    <xs:sequence>
        <xs:element name="package" type="xs:IDREF"
        maxOccurs="unbounded"/>
        <xs:element name="variable" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="new_lock" type="lock_type"/>
    </xs:sequence>
</xs:complexType>
<!-- The lock_type supports workflow data patterns 29
and 30 -->
<xs:simpleType name="lock_type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="read_lock"/>
        <xs:enumeration value="write_lock"/>
        <xs:enumeration value="exclusive_lock"/>
        <xs:enumeration value="none"/>
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="lock_definition_type">
    <xs:sequence>
        <xs:element name="lock_owner" type="xs:IDREFS"
        maxOccurs="unbounded"/>
        <xs:element name="lock" type="lock_type"/>
        <!-- A lock owner may be a resource or a modelling
        construct, e.g., a scope, a process, etc. -->
    </xs:sequence>
</xs:complexType>
<!-- Data-definition tags.  -->
<!-- A package of data may either be input or output.  -->
<!--May be used to support worflow data pattern 9 and 10.
States whether a package is input, output, or in_out -->
<xs:simpleType name="data_flow_direction_type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="input"/>
        <xs:enumeration value="output"/>
        <xs:enumeration value="in_output"/>
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="instance_type">
    <xs:sequence>
        <xs:element name="destination_instance"
        type="xs:IDREFS"/>
        <xs:choice>
            <xs:element name="all_instances"
            type="xs:boolean" minOccurs="0"/>
            <xs:element name="instance_number"
            type="xs:integer" minOccurs="0"/>
            <!-- Target all instances of a task
            -->
```

```
            <!-- Target a particular task-instance
            number -->
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<!-- Packages can be either passed from a source or to
a target. -->
<!--A passing destination defines where a package is
passed on to. -->
<!-- Workflow data pattern 13 is supported by defining
a destination case for a package that is valid in
another case. -->
<!-- Also workflow data patterns 11 and 12 are supported
if the a data package passing from task to task is
including a multiple-instance task. Such a task is
located in a parallel_non_sync blok. -->
<xs:complexType name="passing_type">
    <xs:choice>
        <xs:element name="aim_task" type="instance_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aim_block" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aim_scope" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aim_sphere" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aim_case" type="instance_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aim_process" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
</xs:complexType>
<!-- Variables and their values of one package can be
mapped to variables of a destination package. -->
<xs:complexType name="variable_mapping_type">
    <xs:sequence>
        <xs:element name="from_variable_name"
        type="xs:IDREF"/>
        <xs:element name="to_variable_name"
        type="xs:IDREF"/>
        <xs:element name="transformation_function"
        type="xs:string" minOccurs="0"/>
        <!-- A variable that is mapped may first
        be transformed by a function. -->
    </xs:sequence>
</xs:complexType>
<xs:complexType name="package_mapping_type">
    <xs:sequence>
        <xs:element name="aim_package"
        type="xs:IDREF"/>
        <xs:element name="variable_mapping"
        type="variable_mapping_type" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<!-- Data may be visible in one or many cases. -->
<xs:complexType name="case_visibility_type">
    <xs:sequence maxOccurs="unbounded">
        <xs:element name="case_id" type="xs:IDREFS"/>
    </xs:sequence>
</xs:complexType>
<!-- Coverage of workflow data patterns 33 til including 36.
-->
<!--This tag is useful for active nodes to define as a
precondition for enactment the existance of a particular
variable. -->
<xs:complexType name="data_existence_condition_type">
    <xs:sequence>
        <xs:element name="package" type="xs:IDREF"/>
        <xs:element name="variable" type="xs:IDREF"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<!-- Applicable  in active nodes to check the value of a
particular variable.-->
<xs:complexType name="data_value_condition_type">
    <xs:sequence>
        <xs:element name="package" type="xs:IDREF"/>
        <xs:element name="value_check_statement"
```

```
        type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<!-- Below data-package definitions are given for supporting
various data-flow patterns. -->
<!--Workflow data pattern 2 is supported by defining data
package visibility on a block level. -->
<!-- Supports workflow data pattern 9. When no further
explicit assignment definitions given, the third option
of no data passing in workflow data pattern 9 is performed.
That means all lower-level elements are automatically aware
of the data package. If additional assignment tags are used,
then either data passing is performed via a dedicated
data channel or via an integrated control and data channel,
i.e., in the latter case data flows along control flow.-->
<!-- case_visibility supports workflow data pattern 5.
Here it can be defined if a data package is visible in
all current cases of a workflow process. -->
<!-- With case_visibility_range a limitation in accordance
with workflow data pattern 5 is achieved. The set of cases
can be specified where a data package is equally visible.-->
<!-- passing_destination supports workflow data pattern
8 for passing data patterns from task to task. -->
<!--passing_destination is also useful for supporting
workflow data pattern 9 and 10.  When data_type definition
is on block level. Then explicit data passing from a block
level to a contained lower-level element can be performed.
In case of supporting pattern 10, task-level data packages
can be assigned explicitly to the higher level, e.g.,
a block.-->
<!--control_flow_passing is useful for workflow data
pattern 9. When data_types are defined on a block level,
setting control_flow_passing to true means that an integrated
control and data channel is used. That means data flows
from one node to the next along control flow. -->
<!--If workflow_visibility_range is true, a package is
visible in all cases of a process for all active nodes
contained. Supports workflow data pattern 6 -->
<!--If case_visibility is true, a data package is
visibility for all cases. Covers workflow data pattern 5.
Comparable to class variables in OO.-->
<!-- Workflow data pattern 4 is supported  if a data
package is defined for a task that is instantiated several
times, e.g., when a task is part of a parallel_no_sync
block. -->
<!--Workflow data patterns concerned with internal and
external data passing (patterns 8 - 25) can be supported
with the tags passing_destination and passing_origin in
combination with various tags concerned with visibility.
-->
<!--Workflow data patterns 26 and 27 are concerned with
incoming and outgoing data transfer by value. Such support
can be achieved when data packages are defined for respective
tasks and cases. When data needs to be passed from one
task to another by value, a mapping of a variable and value
can first take place to a case package. Subsequently the
variable mapping to a destination task can be repeated
from a case level.  -->
<!--Workflow data patterns 27  is supported by using a
data package on a case level from which variables are
retrieved into a data package on a task level. The data
with their values are placed back into the case data
package when the task is near completion.-->
<!--Workflow data patterns 25 and 26 are supported with
using tags passing_destination and passing_origin.-->
<!--Workflow data patterns 31 and 32 are supported with
using destination_package_mapping and source_package_mapping.
Instead of passing a data package per se, parts of a data
package in one process node are mapped on properties of
another data package at some other location. -->
<!--Workflow data patterns 33 till 39 are covered by
business rules data are not part of the data_type tag. -->
<!--By using sub_level_visibility in combination with
a data_package definition for a control-flow block element,
it can be determined to which lower level the data_package
is visible. For example, if a block has 5 lower levels of
routing elements and level 4 is defined in a
sub_level_visibility tag, then elements located on the
lowest level don't have visibility of the data package,
```

```
i.e., the 5th level below the definition level of the
data package in question. -->
<xs:complexType name="data_type">
   <xs:sequence>
      <xs:element name="data_flow_direction"
      type="data_flow_direction_type"/>
      <xs:element name="workflow_visibility_range"
      type="xs:boolean" minOccurs="0"/>
      <xs:element name="case_visibility" type="xs:boolean"
      minOccurs="0"/>
      <xs:element name="case_visibility_range"
      type="case_visibility_type" minOccurs="0"/>
      <xs:element name="passing_destination"
      type="passing_type" minOccurs="0"/>
      <xs:element name="passing_origin"
      type="passing_type" minOccurs="0"/>
      <xs:element name="control_flow_passing"
      type="xs:boolean" minOccurs="0"/>
      <xs:element name="destination_package_mapping"
      type="package_mapping_type" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="source_package_mapping"
      type="package_mapping_type" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="sub_level_visibility"
      type="xs:integer" minOccurs="0"/>
      <xs:element name="data_package_ref"
      type="xs:IDREF"/>
   </xs:sequence>
</xs:complexType>
<!-- A scope comprises of several active nodes on a process
that share a set of data. It covers workflow data pattern 3
-->
<xs:complexType name="data_scope_type">
   <xs:complexContent>
      <xs:extension base="data_type">
         <xs:sequence>
            <xs:element name="process" type="xs:IDREF"/>
            <xs:element name="active_nodes"
            type="xs:IDREFS" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<!-- The following definition creates packages of data that
may comprise of variables together with their optional values
and documents. -->
<xs:complexType name="data_package_type">
   <xs:sequence>
      <xs:element name="package_id" type="xs:ID"/>
      <xs:element name="var_section"
      type="variables_def_section" minOccurs="0"/>
      <xs:element name="document_section"
      type="list_of_documents" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<!-- Conjoinment perspective supporting types.  -->
<!-- <xs:complexType name="conjoinment_base_type">
   <xs:sequence>
      <xs:element name="destination_URI"
      type="xs:anyURI"/>
   </xs:sequence>
</xs:complexType> -->
<xs:complexType name="send_task_type">
   <xs:complexContent>
      <xs:extension base="task_type">
         <xs:sequence>
            <xs:element name="destination_URI"
            type="xs:anyURI" minOccurs="0"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="send_transition_type">
   <xs:complexContent>
      <xs:extension base="transition_type">
         <xs:sequence>
            <xs:element name="destination_URI"
            type="xs:anyURI"/>
```

```
            </xs:sequence>
          </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="sourcing_sphere_type">
      <xs:sequence>
        <xs:element name="sphere_id" type="xs:ID"/>
        <xs:element name="owner" type="xs:IDREF"
        minOccurs="0"/>
        <xs:element name="description" type="xs:string"
        maxOccurs="unbounded"/>
        <xs:element name="data" type="data_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lock_change"
        type="lock_change_type" minOccurs="0"/>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
        <xs:element name="interface_in"
        type="interface_type"/>
        <xs:element name="interface_out"
        type="interface_type"/>
        <!-- should refer to a party -->
      </xs:sequence>
      <!-- When multi-lateral contracting takes place,
      the consumer process contains multiple Sourcing
      spheres. Each Sourcing sphere in the consumer
      process is complemented by a service providing
      process. In the latter case sphere comprises the
      entire service-provision process. -->
    </xs:complexType>
    <xs:complexType name="any_sequence_type">
      <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attributeGroup
      ref="logistic_attributes"/>
      <xs:attributeGroup
      ref="contextual_information_attributes"/>
    </xs:complexType>
    <xs:complexType name="choice_type">
      <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attributeGroup ref="logistic_attributes"/>
      <xs:attributeGroup
      ref="contextual_information_attributes"/>
    </xs:complexType>
    <xs:complexType name="condition_type">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="true_branch"
        type="true_branch_type"/>
        <xs:element name="false_branch"
        type="false_branch_type"/>
      </xs:choice>
      <xs:attribute name="condition"
      type="xs:string" use="required"/>
      <xs:attribute name="description"
      type="xs:string"/>
      <xs:attributeGroup ref="logistic_attributes"/>
      <xs:attributeGroup
      ref="contextual_information_attributes"/>
    </xs:complexType>
    <xs:complexType name="false_branch_type">
      <xs:sequence>
        <xs:group ref="common_elements"/>
      </xs:sequence>
      <xs:attributeGroup ref="logistic_attributes"/>
      <xs:attributeGroup
      ref="contextual_information_attributes"/>
    </xs:complexType>
    <xs:complexType name="parallel_no_sync_type">
      <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attributeGroup ref="logistic_attributes"/>
      <xs:attributeGroup
```

```
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="parallel_part_sync_type">
    <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="number" type="xs:NMTOKEN"
    use="required"/>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="parallel_part_sync_cancel_type">
    <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="number" type="xs:NMTOKEN"
    use="required"/>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="parallel_sync_type">
    <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="restricted_parallel_sync_type">
    <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="sequence_type">
    <xs:sequence>
        <xs:group ref="common_elements"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<!--data_existence_precondition supports workflow data
pattern 34.-->
<!--data_existence_postcondition supports workflow data
pattern 36.-->
<!--data_value_precondition supports workflow data pattern
35.-->
<!--data_value_postcondition supports workflow data pattern
37.-->
<xs:complexType name="transition_type">
    <xs:sequence>
        <xs:element name="event" type="event_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="data" type="data_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lock_change"
        type="lock_change_type" minOccurs="0"/>
        <xs:element name="applied_rules" type="xs:IDREFS"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="data_existence_precondition"
        type="data_existence_condition_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="data_value_precondition"
        type="data_value_condition_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="data_existence_postcondition"
        type="data_existence_condition_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="data_value_postcondition"
```

```
        type="data_value_condition_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <!-- Support of workflow data pattern 1. Data block
        visibility for the  task level.-->
        <!-- this tag covers pattern 1 of the workflow data
        patterns. That way data visibility is realized on a
        task level. -->
    </xs:sequence>
    <xs:attribute name="active_node_id" type="xs:ID"
    use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="address" type="xs:string"
    use="optional"/>
</xs:complexType>
<!-- interface_type is relevant for defining the interfaces
for supporting a black box pattern -->
<xs:complexType name="interface_type">
    <xs:complexContent>
        <xs:extension base="transition_type">
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="task_type">
    <xs:complexContent>
        <xs:extension base="transition_type">
            <xs:attribute name="owner" type="xs:IDREF"
            use="optional"/>
            <xs:attribute name="executor"
            type="xs:IDREF" use="optional"/>
            <xs:attribute name="responsible"
            type="xs:IDREF" use="optional"/>
            <xs:attribute name="necessary_resources"
            type="xs:IDREF" use="optional"/>
            <xs:attribute name="result" type="xs:string"/>
            <xs:attribute name="notify" type="xs:string"/>
            <xs:attribute name="enabled" type="state_type"/>
            <xs:attribute name="start_date"
            type="xs:date" use="optional"/>
            <xs:attribute name="start_time"
            type="xs:time" use="optional"/>
            <xs:attribute name="end_date" type="xs:date"
            use="optional"/>
            <xs:attribute name="end_time" type="xs:time"
            use="optional"/>
            <xs:attributeGroup ref="logistic_attributes"/>
            <xs:attributeGroup
            ref="contextual_information_attributes"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="terminate_type"/>
<xs:complexType name="timeout_type">
    <xs:sequence>
        <xs:group ref="common_elements" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="time" type="xs:string" use="required"/>
    <xs:attribute name="type" default="absolute">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="relative"/>
                <xs:enumeration value="s_relative"/>
                <xs:enumeration value="absolute"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="true_branch_type">
    <xs:sequence>
        <xs:group ref="common_elements"/>
    </xs:sequence>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="wait_all_type">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="event_ref" type="xs:IDREF"/>
        <xs:element name="timeout" type="timeout_type"/>
```

```
    </xs:choice>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="wait_any_type">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="event_ref" type="xs:IDREF"/>
        <xs:element name="timeout" type="timeout_type"/>
    </xs:choice>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="while_do_type">
    <xs:sequence>
        <xs:group ref="common_elements"/>
    </xs:sequence>
    <xs:attribute name="condition" type="xs:string"
    use="required"/>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
</xs:complexType>
<xs:complexType name="route">
    <xs:sequence>
        <xs:group ref="common_elements"/>
        <xs:element name="data_scope"
        type="data_scope_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <!-- A Data_scope comprises of several active
        nodes, i.e., task, transition, and all active
        conjoinment nodes.  Data_scopes cover workflow
        data pattern 3.-->
    </xs:sequence>
    <xs:attribute name="tag_name" type="xs:string"
    use="required"/>
    <xs:attribute name="process_id" type="xs:ID"
    use="required"/>
    <xs:attribute name="enabled" type="state_type"/>
    <xs:attribute name="created_by" type="xs:string"/>
    <xs:attribute name="date" type="xs:string"/>
    <xs:attributeGroup ref="logistic_attributes"/>
    <xs:attributeGroup
    ref="contextual_information_attributes"/>
    <!-- Control over the process and monitoring rights
    and obligations are specified through deontic
    assignments and reaction rules (executive for
    monitoring and copier reaction rules for control
    rights) -->
    <!-- extension -->
</xs:complexType>
<!-- PART 4          Syntax          -->
<!-- In this part, we provide the syntax definitions
of eSML, i.e., the structure of eSML. First, we
provide three basic structures, i.e., the Variable
definition section, Rule definition section, and Process
definition section. In each of these sections can
respectively be defined data constructs, rule constructs,
and process constructs. -->
<xs:complexType name="variables_def_section">
    <xs:sequence maxOccurs="unbounded">
        <xs:choice>
            <xs:element name="string_var"
            type="string_type"/>
            <xs:element name="real_var"
            type="real_type"/>
            <xs:element name="integer_var"
            type="integer_type"/>
            <xs:element name="boolean_var"
            type="boolean_type"/>
            <xs:element name="date_var"
            type="date_type"/>
            <xs:element name="time_var"
            type="time_type"/>
            <xs:element name="event_var"
            type="event_type"/>
            <xs:element name="money_var"
            type="money_type"/>
            <xs:element
            name="external_resource_reference_var"
```

```
            type="external_resource_reference_type"/>
            <xs:element name="list_of_events_var"
            type="list_of_events_type"/>
            <xs:element name="list_of_strings_var"
            type="list_of_strings_type"/>
            <xs:any/>
            <!-- <xs:element
            name="internal_resource_reference_var"
            type="internal_resource_reference_type"/>
            -->
            <!--Supports workflow data pattern 7.
            This way environment data can be brought
            into a data package. -->
            <!-- Required to allow the definition of
            user-defined complex types and lists -->
         </xs:choice>
         <xs:element name="lock_definition"
         type="lock_definition_type" minOccurs="0"/>
         <!-- For  data definitions of a data package
         an initial lock can be placed. -->
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="rule_def_section">
      <xs:sequence maxOccurs="unbounded">
         <xs:choice>
            <xs:element name="state_constraint_rule"
            type="state_constraint_rule_type"
            minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="dynamic_constraint_rule"
            type="dynamic_constraint_rule_type"
            minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="computational_derivation_rule"
            type="computational_derivation_rule_type"
            minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="linguistic_derivation_rule"
            type="linguistic_derivation_rule_type"
            minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="reaction_rule"
            type="reaction_rule_type" minOccurs="0"
            maxOccurs="unbounded"/>
            <xs:element name="free_text_rule"
            type="free_text_rule_type" minOccurs="0"
            maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <!-- Next, the monitorability patterns are defined.-->
   <xs:complexType name="sink_type">
      <xs:sequence maxOccurs="unbounded">
         <xs:element name="provider_sphere"
         type="xs:IDREF"/>
         <xs:element name="provider_active_node"
         type="xs:IDREF"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="link_properties">
      <xs:sequence>
         <xs:element name="consumer_sphere"
         type="xs:IDREF"/>
         <xs:element name="consumer_active_node"
         type="xs:IDREF"/>
         <xs:element name="provider"
         type="sink_type"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="enactment_properties">
      <xs:sequence>
         <xs:element name="consumer_sphere"
         type="xs:IDREF"/>
         <xs:element name="provider_sphere"
         type="xs:IDREF"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="lifecycle_link_properties">
      <xs:sequence>
         <xs:element name="consumer_sphere"
         type="xs:IDREF"/>
         <xs:element name="consumer_active_node"
         type="xs:IDREF"/>
```

```
      <xs:element name="consumer_lifecycle_node"
      type="xs:IDREF"/>
      <xs:element name="provider_sphere"
      type="xs:IDREF"/>
      <xs:element name="provider_active_node"
      type="xs:IDREF"/>
      <xs:element name="provider_lifecycle_node"
      type="xs:IDREF"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="lifecycle_polling_patterns">
    <xs:sequence maxOccurs="unbounded">
      <xs:choice>
        <xs:element
        name="lifecycle_transition_polling"
        type="lifecycle_link_properties"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element
        name="lifecycle_state_polling"
        type="lifecycle_link_properties"
        minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="lifecycle_messaging_patterns">
    <xs:sequence maxOccurs="unbounded">
      <xs:choice>
        <xs:element
        name="lifecycle_transition_messaging"
        type="lifecycle_link_properties"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element
        name="lifecycle_state_messaging"
        type="lifecycle_link_properties"
        minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="polling_patterns">
    <xs:choice maxOccurs="unbounded">
      <xs:element name="enactment_takeover"
      type="enactment_properties" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="termination_takeover"
      type="enactment_properties" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="transition_polling"
      type="link_properties" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="lifecycle_polling"
      type="lifecycle_polling_patterns"
      minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="messaging_patterns">
    <xs:sequence maxOccurs="unbounded">
      <xs:choice>
        <xs:element name="enactment_propagation"
        type="enactment_properties" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="termination_propagation"
        type="enactment_properties" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="transition_messaging"
        type="link_properties" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="lifecycle_messaging"
        type="lifecycle_messaging_patterns"
        minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="monitorability_patterns">
    <xs:sequence>
      <xs:element name="polling" type="polling_patterns"
      minOccurs="0"/>
      <xs:element name="messaging"
      type="messaging_patterns" minOccurs="0"/>
    </xs:sequence>
```

```
</xs:complexType>
<!-- The definition of life cycles is given. Life cycles may
be present for  a process level, or on a task level. A life
cycle has control-flow too.-->
<xs:complexType name="lifecycle_node_type">
   <xs:attribute name="name" type="xs:ID" use="required"/>
   <xs:attribute name="tag_name" type="xs:string" u
   se="required"/>
</xs:complexType>
<xs:complexType name="lifecycle_sequence_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<!-- A nesting state contains further life-cycle nodes. -->
<xs:complexType name="nesting_state_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="lifecycle_parallel_sync_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="lifecycle_parallel_no_sync_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="lifecycle_parallel_part_sync_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="number" type="xs:NMTOKEN"
   use="required"/>
</xs:complexType>
<xs:complexType
name="lifecycle_parallel_part_sync_cancel_type">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"
      maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="number" type="xs:NMTOKEN"
   use="required"/>
</xs:complexType>
<!-- Below, all elements of a life cycle are grouped
together. -->
<xs:group name="lifecycle_elements">
   <xs:choice>
      <xs:element name="transition"
      type="lifecycle_node_type"/>
      <xs:element name="nesting_state"
      type="nesting_state_type"/>
      <xs:element name="atomic_state"
      type="lifecycle_node_type"/>
      <xs:element name="lifecycle_sequence"
      type="lifecycle_sequence_type"/>
      <xs:element name="lifecycle_parallel_sync"
      type="lifecycle_parallel_sync_type"/>
      <xs:element name="lifecycle_parallel_no_sync"
      type="lifecycle_parallel_no_sync_type"/>
      <xs:element name="lifecycle_parallel_part_sync"
      type="lifecycle_parallel_part_sync_type"/>
      <xs:element name="lifecycle_parallel_part_sync_cancel"
      type="lifecycle_parallel_part_sync_cancel_type"/>
   </xs:choice>
</xs:group>
<xs:complexType name="lifecycle_details">
   <xs:sequence>
      <xs:group ref="lifecycle_elements"/>
   </xs:sequence>
   <xs:attribute name="process_id" type="xs:IDREF"/>
</xs:complexType>
```

```
<xs:complexType name="lifecycles">
   <xs:sequence>
      <xs:element name="process_lifecycle"
      type="lifecycle_details" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="active_node_lifecycle"
      type="lifecycle_details" minOccurs="0"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<!-- Next, the mapping of life-cycle stages is defined-->
<xs:complexType name="lifecycle_mapping_details">
   <xs:complexContent>
      <xs:extension base="link_properties">
         <xs:attribute name="mapping_name"
         type="xs:ID" use="required"/>
         <xs:attribute name="node_type">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:pattern
                  value="lifecycle_transition|
                  lifecycle_state"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="mapping_details">
   <xs:sequence>
      <xs:element name="process_lifecycle_mapping"
      type="lifecycle_mapping_details" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="active_node_lifecycle_mapping"
      type="lifecycle_mapping_details" minOccurs="0"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="provider_type">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="provider_process"
      type="xs:IDREF"/>
      <xs:element name="provider_active_node"
      type="xs:IDREF"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="active_node_label_mapping_type">
   <xs:sequence>
      <xs:element name="consumer_process"
      type="xs:IDREF"/>
      <xs:element name="consumer_active_node"
      type="xs:IDREF"/>
      <xs:element name="provider_process"
      type="xs:IDREF"/>
      <xs:element name="provider_active_node"
      type="xs:IDREF"/>
      <xs:element name="provider"
      type="provider_type"/>
   </xs:sequence>
</xs:complexType>
<!-- Next, the resource-perspective definition for
covering the organizational aspect. -->
<!-- An organizational unit and its subclasses is
defined. These subclasses are permanent_organizational_unit
and temporary_organizational_unit -->
<xs:group name="organizational_unit_elements">
   <xs:sequence>
      <xs:element name="name" type="xs:ID"/>
      <xs:element name="start_date" type="xs:date"
      maxOccurs="unbounded"/>
      <xs:element name="description" type="xs:string"
      maxOccurs="unbounded"/>
      <xs:element name="business_objectives"
      type="xs:string" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="temporary_organizational_unit_type">
   <xs:complexContent>
      <xs:extension base="organizational_unit_type">
```

```
            <xs:sequence>
               <xs:element name="end_date" type="xs:date"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="permanent_organizational_unit_type">
      <xs:complexContent>
         <xs:extension base="organizational_unit_type"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="organizational_unit_type">
      <xs:sequence>
         <xs:group ref="organizational_unit_elements"/>
         <xs:element name="resource_type" type="xs:IDREF"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="resource_nref"
         type="resource_nref_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="collection" type="xs:IDREF"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="individual_resource"
         type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="organizational_position_type">
      <xs:sequence>
         <xs:element name="description" type="xs:string"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="organizational_unit"
         type="xs:IDREF" maxOccurs="unbounded"/>
         <xs:element name="actor" type="xs:IDREF"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="privilege" type="xs:IDREF"
         minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <!-- Several resource_type may be part of organizational_units.
   A resource_type has subclasses, namely a role, a machine,
   space, and production_material. Furthermore, a resource_type
   is referred by a typed_collection. -->
   <xs:group name="resource_perspective">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:element name="machine_type" type="machine_type_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="production_material_type"
         type="production_material_type_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="space_type" type="space_type_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="capability" type="capability_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="privilege" type="privilege_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="power" type="power_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="power_delegation"
         type="power_delegation_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="concrete_collection"
         type="concrete_collection_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="mixed_collection"
         type="mixed_collection_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="typed_collection"
         type="typed_collection_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="resource_nref"
         type="resource_nref_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="available" type="available_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="space" type="space_type"
         minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="production_material"
         type="production_material_type" minOccurs="0"
         maxOccurs="unbounded"/>
         <xs:element name="containment_type"
```

```
        type="containment_type_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="machine" type="machine_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="capacity" type="capacity_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="rate_of_usage"
        type="rate_of_usage_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="actor" type="actor_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="role_delegation"
        type="role_delegation_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="appointment"
        type="appointment_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="correlation"
        type="correlation_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="hierarchy_relationship"
        type="hierarchy_relationship_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="temporary_organizational_unit"
        type="temporary_organizational_unit_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="permanent_organizational_unit"
        type="permanent_organizational_unit_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="organizational_position"
        type="organizational_position_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="role" type="role_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="connection_relationship"
        type="connection_relationship_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="resource_type"
        type="resource_type_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="collection"
        type="collection_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="individual_resource"
        type="individual_resource_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="non_actor" type="non_actor_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="consumable_resource"
        type="consumable_resource_type" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="organizational_unit"
        type="organizational_unit_type" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:choice>
</xs:group>
<xs:group name="resource_type_elements">
    <xs:sequence>
        <xs:element name="name" type="xs:ID"/>
        <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="resource_type_type">
    <xs:sequence>
        <xs:group ref="resource_type_elements"/>
        <xs:element name="typed_collection" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="machine_type_type">
    <xs:complexContent>
        <xs:extension base="resource_type_type"/>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="production_material_type_type">
    <xs:complexContent>
        <xs:extension base="resource_type_type">
            <xs:sequence>
```

```
            <xs:element name="production_material"
            type="xs:IDREF"/>
          </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="space_type_type">
    <xs:complexContent>
        <xs:extension base="resource_type_type">
            <xs:sequence>
                <xs:element name="space" type="xs:IDREF"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!--A role definition  and all related entities.-->
<xs:group name="role_elements">
    <xs:sequence>
        <xs:element name="qualifications_required"
        type="xs:string" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element name="skills_required" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="role_type">
    <xs:complexContent>
        <xs:extension base="resource_type_type">
            <xs:sequence>
                <xs:group ref="role_elements"/>
                <xs:element name="power" type="xs:IDREF"
                minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="capability"
                type="xs:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
                <!-- this is refering to required
                capabilities for a certain role -->
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:group name="feature_elements">
    <xs:sequence>
        <xs:element name="name" type="xs:ID"/>
        <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="capability_type">
    <xs:sequence>
        <xs:group ref="feature_elements"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="privilege_type">
    <xs:sequence>
        <xs:group ref="feature_elements"/>
        <xs:element name="role" type="xs:IDREF"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="power_type">
    <xs:sequence>
        <xs:group ref="feature_elements"/>
        <xs:element name="capability" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="privilege" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="power_delegation_type">
    <xs:sequence>
        <xs:group ref="timing_elements"/>
        <xs:element name="superior_role" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="inferior_role" type="xs:IDREF"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="power" type="xs:IDREF"
        minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

```
<!--The definition of a collection and all subclasses.
These subclasses are mixed_collection, concrete_collection,
and typed_collection -->
<xs:group name="collection_elements">
   <xs:sequence>
      <xs:element name="name" type="xs:ID"/>
      <xs:element name="description" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="collection_type">
   <xs:sequence>
      <xs:group ref="collection_elements"/>
      <xs:element name="organizational_unit"
      type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="concrete_collection_type">
   <xs:complexContent>
      <xs:extension base="collection_type">
         <xs:sequence>
            <xs:element name="individual_resource"
            type="xs:IDREF" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="mixed_collection_type">
   <xs:complexContent>
      <xs:extension base="collection_type">
         <xs:sequence>
            <xs:element name="resource_nref"
            type="resource_nref_type" maxOccurs="unbounded"/>
            <xs:element name="individual_resource"
            type="xs:IDREF" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="typed_collection_type">
   <xs:complexContent>
      <xs:extension base="collection_type">
         <xs:sequence>
            <xs:element name="resource_nref_type"
            type="resource_nref_type" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="resource_nref_type">
   <xs:sequence>
      <xs:element name="resource_type_ref"
      type="xs:IDREF"/>
      <xs:element name="number" type="xs:integer"/>
      <xs:element name="typed_collection_reference"
      type="xs:IDREF"/>
      <xs:element name="mixed_collection_reference"
      type="xs:IDREF"/>
   </xs:sequence>
</xs:complexType>
<!-- Resource related definitions.  -->
<xs:group name="individual_resource_elements">
   <xs:sequence>
      <xs:element name="name" type="xs:ID"/>
      <xs:element name="address" type="xs:string"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="individual_resource_type">
   <xs:sequence>
      <xs:group ref="individual_resource_elements"/>
      <xs:element name="available" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="concrete_collection"
      type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="mixed_collection"
      type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="organizational_unit"
      type="xs:IDREF" maxOccurs="unbounded"/>
```

```
      <xs:element name="capability" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:group name="available_type_elements">
   <xs:sequence>
      <xs:element name="number" type="xs:ID"/>
      <xs:element name="start_date" type="xs:date"
      minOccurs="0"/>
      <xs:element name="start_time" type="xs:time"
      minOccurs="0"/>
      <xs:element name="end_date" type="xs:date"
      minOccurs="0"/>
      <xs:element name="end_time" type="xs:time"
      minOccurs="0"/>
      <xs:element name="status" type="xs:string"
      minOccurs="0"/>
      <xs:element name="reserved_for" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="available_type">
   <xs:sequence>
      <xs:group ref="available_type_elements"/>
      <xs:element name="individual_resource"
      type="xs:IDREF" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<!-- non_actor definition with their subclasses. -->
<xs:group name="non_actor_elements">
   <xs:sequence>
      <xs:element name="description" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="contact" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="phone" type="xs:integer"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="non_actor_type">
   <xs:complexContent>
      <xs:extension base="individual_resource_type">
         <xs:sequence>
            <xs:group ref="non_actor_elements"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:group name="space_elements">
   <xs:sequence>
      <xs:element name="building_number"
      type="xs:string"/>
      <xs:element name="room_number"
      type="xs:string"/>
      <xs:element name="description"
      type="xs:string" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="capacity" type="xs:string"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="space_type">
   <xs:complexContent>
      <xs:extension base="non_actor_type">
         <xs:sequence>
            <xs:group ref="space_elements"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="consumable_resource_type">
   <xs:complexContent>
      <xs:extension base="non_actor_type"/>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="production_material_type">
   <xs:complexContent>
      <xs:extension
      base="consumable_resource_type"/>
   </xs:complexContent>
```

```
</xs:complexType>
<xs:complexType name="containment_type_type">
   <xs:sequence>
      <xs:element name="superior_production_material"
      type="xs:IDREF"/>
      <xs:element name="inferior_production_material"
      type="xs:IDREF"/>
      <xs:element name="description" type="xs:string"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="connection_relationship_type">
   <xs:sequence>
      <xs:element name="source_production_material"
      type="xs:IDREF"/>
      <xs:element name="target_production_material"
      type="xs:IDREF"/>
      <xs:element name="description" type="xs:string"
      maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="machine_type">
   <xs:complexContent>
      <xs:extension base="consumable_resource_type">
         <xs:sequence>
            <xs:element name="number" type="xs:string"
            maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:group name="capacity_elements">
   <xs:sequence>
      <xs:element name="amount" type="xs:decimal"/>
      <xs:element name="unit" type="xs:string"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="capacity_type">
   <xs:sequence>
      <xs:group ref="capacity_elements"/>
      <xs:element name="machine" type="xs:IDREF"/>
      <xs:element name="production_material" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:group name="rate_of_usage_elements">
   <xs:sequence>
      <xs:element name="usage_quantity" type="xs:decimal"/>
      <xs:element name="usage_period" type="xs:decimal"/>
      <xs:element name="period_unit" type="xs:string"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="rate_of_usage_type">
   <xs:sequence>
      <xs:group ref="rate_of_usage_elements"/>
      <xs:element name="machine" type="xs:IDREF"/>
      <xs:element name="production_material" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<!-- actor definition with their subclasses. -->
<xs:group name="actor_elements">
   <xs:sequence>
      <xs:element name="actor_id" type="xs:ID"/>
      <xs:element name="first_name" type="xs:string"/>
      <xs:element name="last_name" type="xs:string"/>
      <xs:element name="email" type="xs:string"
      maxOccurs="unbounded"/>
      <xs:element name="login_id" type="xs:string"/>
      <xs:element name="phone" type="xs:string"
      maxOccurs="unbounded"/>
      <xs:element name="qualification" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="skill" type="xs:string" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="previous_work_experience"
      type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="held_responsibility" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
```

```
    </xs:sequence>
</xs:group>
<xs:complexType name="actor_type">
    <xs:sequence>
        <xs:group ref="actor_elements"/>
        <xs:element name="role" type="xs:IDREF" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:group name="timing_elements">
    <xs:sequence>
        <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="start_date" type="xs:date"
        minOccurs="0"/>
        <xs:element name="start_time" type="xs:time"
        minOccurs="0"/>
        <xs:element name="end_date" type="xs:date"
        minOccurs="0"/>
        <xs:element name="end_time" type="xs:time"
        minOccurs="0"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="role_delegation_type">
    <xs:sequence>
        <xs:group ref="timing_elements"/>
        <xs:element name="source_actor" type="xs:IDREF"/>
        <xs:element name="target_actor" type="xs:IDREF"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="appointment_type">
    <xs:sequence>
        <xs:group ref="timing_elements"/>
        <xs:element name="actor" type="xs:IDREF"
        maxOccurs="unbounded"/>
        <xs:element name="task" type="xs:IDREF"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<!-- The relationships of organizational_unit entities
to each other is established. That includes hierarchy
relationships in terms of superior and inferior, and
a correlation of one organizational_unit to another one,
e.g., a relationship exists because bith are involved
in the same project or in the same business process,
etc.-->
<xs:complexType name="correlation_type">
    <xs:sequence>
        <xs:element name="description" type="xs:string"
        maxOccurs="unbounded"/>
        <xs:element name="related_to" type="xs:IDREF"/>
        <xs:element name="relation_source" type="xs:IDREF"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="hierarchy_relationship_type">
    <xs:sequence>
        <xs:element name="description" type="xs:string"
        maxOccurs="unbounded"/>
        <xs:element name="superior" type="xs:IDREF"/>
        <xs:element name="inferior" type="xs:IDREF"/>
    </xs:sequence>
</xs:complexType>
<!-- Next, the process definition wraps up all respective,
available elements. -->
<xs:complexType name="process_def_section">
    <xs:sequence>
        <xs:element name="process" type="route"
        maxOccurs="unbounded"/>
        <xs:element name="lifecycle_definitions"
        type="lifecycles" minOccurs="0"/>
        <xs:element name="lifecycle_mappings"
        type="mapping_details" minOccurs="0"/>
        <xs:element name="active_node_label_mapping"
        type="active_node_label_mapping_type"
        minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="monitorability"
        type="monitorability_patterns" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

```
<!-- Next, we define the possible combinations of data
constructs, rule constructs and process constructs that
can be used in an e-contract sub-structure.  -->
<xs:complexType name="only_vars_section">
   <xs:sequence>
      <xs:element name="var_section"
      type="variables_def_section"/>
      <xs:element name="process_section"
      type="process_def_section" minOccurs="0"/>
      <xs:element name="snippet_section"
      type="snippet_type" minOccurs="0"/>
      <!-- required to support the inclusion of
      externally defined data items. -->
   </xs:sequence>
</xs:complexType>
<xs:complexType name="vars_and_processes_section">
   <xs:sequence>
      <xs:element name="var_section"
      type="variables_def_section" minOccurs="0"/>
      <xs:element name="process_section"
      type="process_def_section" minOccurs="0"/>
      <xs:element name="snippet_section"
      type="snippet_type" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="all_section">
   <xs:sequence>
      <xs:element name="var_section"
      type="variables_def_section" minOccurs="0"/>
      <xs:element name="rule_section"
      type="rule_def_section" minOccurs="0"/>
      <xs:element name="process_section"
      type="process_def_section" minOccurs="0"/>
      <xs:element name="snippet_section"
      type="snippet_type" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<!-- definition of contract sub-structures of
the 4W structure-->
<xs:complexType name="resource_section_type">
   <xs:sequence>
      <xs:group ref="resource_perspective"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="company_info">
   <xs:sequence>
      <xs:element name="company_data"
      type="only_vars_section"/>
      <xs:element name="company_contact_data"
      type="only_vars_section"/>
      <xs:element name="resource_section"
      type="resource_section_type" minOccurs="0"/>
      <xs:element name="context_section"
      type="only_vars_section" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="local_language"
   type="xs:string"/>
</xs:complexType>
<xs:complexType name="value_types">
   <xs:choice>
      <xs:element name="product"
      type="vars_and_processes_section"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="service"
      type="vars_and_processes_section"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="financial_reward"
      type="vars_and_processes_section"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
</xs:complexType>
<xs:complexType name="list_of_data_packages">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="data_package"
      type="data_package_type"/>
   </xs:sequence>
</xs:complexType>
<!-- Finally, we define the root element of
```

```
    the schema and the complete contract
    structure -->
    <xs:element name="contract">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="party"
          type="company_info"
          maxOccurs="unbounded"/>
          <xs:element name="mediator"
          type="company_info" minOccurs="0"
          maxOccurs="unbounded"/>
          <xs:element name="data_definition_section"
          type="list_of_data_packages" minOccurs="0"/>
          <xs:element name="business_context_provisions"
          type="all_section" minOccurs="0"
          maxOccurs="unbounded"/>
          <xs:element name="legal_context_provisions"
          type="all_section" minOccurs="0"
          maxOccurs="unbounded"/>
          <xs:element name="other_context_provisions"
          type="all_section" minOccurs="0"
          maxOccurs="unbounded"/>
          <xs:element name="exchanged_value"
          type="value_types" minOccurs="2"
          maxOccurs="unbounded"/>
          <xs:element name="exchange_provisions"
          type="all_section" minOccurs="2"
          maxOccurs="unbounded"/>
          <xs:element name="agreed_ontology"
          type="external_resource_reference_type"
          maxOccurs="unbounded"/>
          <!--WHO Section contains the description
          of the parties and the possible mediators.
          -->
          <!--WHERE Section contains the description
          of the context of the agreement. -->
          <!--WHAT Section contains the description
          of the exchanged goods/services and the
          conditions for the exchange. -->
        </xs:sequence>
        <xs:attribute name="contract_id" type="xs:ID"/>
        <xs:attribute name="global_language" type="xs:string"/>
        <xs:attribute name="web_service_uri" type="xs:string"/>
      </xs:complexType>
    </xs:element>
</xs:schema>
```