Averiti Notation Description

Domain Model Information Requirements

This paper describes the elements and concepts which form a domain model using the Averiti notation. These include Component Information, Structural Information, Functional Information and Operational Information. Each is described in more detail in the sections below.

Component Information

The basic elements and concepts which form an Averiti domain model:

- **System**: a technical system which accomplishes tasks on demand. All systems are processors of something. Human actors in a system of systems can also be described as systems.
- **Function**: the function performed by a system.
- **Media**: Systems transmit their effects through various media which themselves have properties (e.g. pressure, temperature). These media include:
  - **Gases**
  - **Liquids**
  - **Mechanical movement**
  - **Electric current**
  - **Control signals** (digital or analogue)
- **Generator**: A system which changes a media qualitatively into something else (e.g. an engine changing fuel (liquid) into rotary movement).
- **Transformer**: A system which changes only some parameters of something (e.g. a pump changing low pressure gas to high pressure gas).
- **instance_of**: The system is an instance of a particular generic system from the domain model library (e.g. instance_of diesel_engine).

Structural Information

Structural information allows the structure of a system of systems or a system and it subsystems to be described. This information is critical for analysing queries against the model. For example it should be able to answer questions such as which subsystem components are part of a particular subsystem? Or are two specific subsystems part of the Solar Array Subsystem? This level of detail supports hypotheses of new facts which would make the reported facts into a coherent whole. Query rules can be applied against structural information, for example "Solar Array Subsystem must contain Solar Cells". Models can be tested against such rules and variances flagged up for analysis and rectification.

Structural information includes the following elements:

- **part_of**: a pointer up in the system hierarchy; the subsystem belongs to the system/subsystem to which the part_of notation refers.
- **made_of**: a pointer down in the system hierarchy; the subsystem to which the made_of notation refers is a sub-component of the system/subsystem.

- **attached_from**: a pointer across the system hierarchy; the subsystem to which the attached_from refers is attached from the subsystem.
- **attached_to**: a pointer across the system hierarchy; the subsystem is attached to the subsystem to which attached_to refers.
- **adjacent_from**: a pointer across the system hierarchy; the subsystem receives an input from the subsystem to which adjacent_from refers.
- **adjacent_to**: a pointer across the system hierarchy; the subsystem provides an input to the subsystem to which the subsidiary element adjacent_to_name refers.
- **adjacent_to_name**: a subsidiary element of adjacent_to; specifies the name of the subsystem that the subsystem is adjacent_to.
- **function**: a subsidiary element of adjacent_to; see Functional Information below.
- **media**: a subsidiary element of adjacent_to; see Functional Information below.
- **caption**: a subsidiary element of adjacent_to; see Functional Information below.

The structural information defined above also lets a model be queried for completeness. If a subsystem is detailed as being adjacent_to another subsystem, but that subsystem either does not exist in the model or does not hold the equivalent attached_from relationship then the model is incomplete.

Functional Information

Functional information provides further detail on the specific function of the subsystems and the media on which they operates. This is the information about the role which individual subsystems play when the equipment is operating normally. Its significance parallels that of the structural information, i.e. it is essential for analysing queries, hypothesizing new facts and checking equipment element features. If we consider the Feed Mechanism of the Ion Thruster, we can identify its primary function, its functional dependencies and the functional capabilities it provides to other subsystems.

Functional information includes the following elements:

- **primary_function**: the specific function of the system. This can have a number of element values or predicates, each of which can be further expanded by the user to provide more information when appropriate:
  - **contain**: primary_function is to contain a substance (e.g. a fuel tank containing liquid) or contain another subsystem; expansion possibilities include contain_propellant, contain_gas, etc.
  - **drive**: primary function is to mechanically drive another subsystem (e.g. an engine providing rotation (media) to drive a gear box); expansion possibilities include drive_shaft.
  - **lubricate**: primary function is to lubricate another subsystem.
  - **provide**: primary function is to provide a substance to another subsystem; expansion possibilities include provide_power, provide_propellant, etc.
  - **control**: primary function is to control the function of another subsystem or the flow of a substance; expansion possibilities include control_start, control_stop; control_regulate, etc.
  - **emit**: primary function is to emit a substance; expansion possibilities include emit_electrons, emit_vapour, etc.
  - **alarm**: primary function is to raise an alert; a special predicates.

    - **measure**: primary function is to measure another subsystem's function or a media; a special predicate.
- **function**: the function provided to an adjacent_to subsystem.
- **media**: the media on which the subsystem transmits its effects (e.g. liquid) to an adjacent_to subsystem.
- **caption**: more specific information on the media (e.g. the liquid is specifically diesel fuel).

Operational Information

Provides operational information about the current status of the system in performing its function. This operational information can be used to support ascertaining relationships between reported facts, e.g. can a system-level problem be related back to the status of a subsystem. The ordering of subsystem status updates can also be determined through the model, based on the functional dependencies defined in the functional information. Thus if Subsystem a.1.2 is dependent from an input from Subsystem a.1.1, if Subsystem a.1.1 is determined to have failed at one step in the simulation, at the next step Subsystem a.1.2 will also fail. In this way fail states can be traced through the model.

Operational information includes the following elements:

- **status**: GREEN/AMBER/RED equating to OK/Damaged/Failed

Example Subsystem Instance

Based on the attributes and values detailed above, an example subsystem instance can be shown. The example used is an instance of the Electron_Emitter subsystem prototype. Its instanced attributes and values, together with explicatory text is shown below:

subsystem_name = Electron_Emitter01

The specific name of this Electron_Emitter instance - in this case "Electron_Emitter01".

instance_of = Spacecraft.Electrostatic_Thruster.Electron_Emitter

This subsystem has been instanced from the Spacecraft Subsystem Library subsystem prototype "Electron_Emitter".

number_made_of = 0

In the model the Electron Emitter prototype is a simple node, that is, it has not been decomposed into lower level subsystem. The number_made_of attribute is therefore set to 0.

made_of = attribute not included in subsystem instance.

The Electron Emitter subsystem is a simple node, that is, it has not been decomposed into lower level subsystems. The made_of attribute is therefore not included in the instance.

number_part_of = 1

The Electron Emitter subsystem is part of only a single higher level subsystem, number_part_of is therefore set to 1.

part_of = Electrostatic_Thruster01

The Electron Emitter subsystem is part of the higher level Electrostatic_Thruster01 subsystem.

primary_function = emit_electrons

Describes the primary function of the subsystem, in this case emit electrons.

subsystem_notes = A device which ensures electron current discharged from vehicle to prevent charge build up

Notes which provide additional human readable information on the function and/or operation of the subsystem.

number_attached_to = 1

The subsystem is attached to one other subsystem in the model, therefore the attribute is set to 1.

attached_to = Primary_Structure01

The subsystem is attached to the Primary_Structure01 specific subsystem.

number_adjacent_from = 1

The subsystem is adjacent from (i.e. receives an input from) to one other subsystem in the model, therefore the attribute is set to 1.

adjacent_from = Power_Bus01

The subsystem is adjacent from the Power_Bus01 specific subsystem.

number_adjacent_to = 1

The subsystem is adjacent to (i.e. it provides an input to) to one other subsystem in the model, therefore the attribute is set to 1.

adjacent_to has a number of sub-attributes as follows:

adjacent_to_name = Ionization_Device01

The subsystem is adjacent to the Ionization_Device01 specific subsystem.

function = provide

media = electric_charge

caption = electric_charge

The subsystem provides electric charge to Ionization_Device01.

status = GREEN

The current status of the subsystem is green, i.e. fully functional.

System Information

A system, or higher level subsystem, is defined as a collection of subsystems. A system is simply defined as a formal list of the subsystems that comprise the system. No other structural or functional information is required, because that is all defined by the subsystem links.

XML Notation

XML Subsystems and Prototypes

Subsystems and subsystem prototypes are captured in the Averiti notation language as XML documents. This allows them to be easily machine and human readable and allows them to be easily shared and acted upon if required by other applications.

A single schema, subsystem.xsd, is used to capture both subsystems and subsystem prototypes. It is presented in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="subsystem">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="subsystem_name" type="xs:string"/>
      <xs:element name="instance_of" type="xs:string"/>
      <xs:element name="number_made_of" type="xs:integer"/>
      <xs:element name="made_of" type="xs:string" minOccurs="0"/>
      <xs:element name="number_part_of" type="xs:integer"/>
      <xs:element name="part_of" type="xs:string" minOccurs="0"/>
      <xs:element name="primary_function" type="xs:string"/>
      <xs:element name="subsystem_notes" type="xs:string"/>
      <xs:element name="number_attached_to" type="xs:integer"/>
      <xs:element name="attached_to" type="xs:string" minOccurs="0"/>
      <xs:element name="number_adjacent_from" type="xs:integer"/>
      <xs:element name="adjacent_from" type="xs:string" minOccurs="0"/>
      <xs:element name="number_adjacent_to" type="xs:integer"/>
      <xs:element name="adjacent_to" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="function" type="xs:string"/>
            <xs:element name="media" type="xs:string"/>
            <xs:element name="caption" type="xs:string"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element name="status" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Figure 1: subsystem.xsd

The Electron_Emitter prototype example used previously is expressed in the Averiti XML notation at Figure 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Edited by Averiti -->
<subsystem>
    <subsystem_name>Electron_Emitter</subsystem_name>
    <instance_of>prototype</instance_of>
    <number_made_of>0</number_made_of>
    <number_part_of>1</number_part_of>
    <part_of>Electrostatic_Thruster</part_of>
    <primary_function>emit_electrons</primary_function>
    <subsystem_notes>A device which ensures electron current discharged from vehicle to prevent charge build up</subsystem_notes>
    <number_attached_to>1</number_attached_to>
    <attached_to>Primary_Structure</attached_to>
    <number_adjacent_from>1</number_adjacent_from>
    <adjacent_from>Power_Bus</adjacent_from>
    <number_adjacent_to>1</number_adjacent_to>
    <adjacent_to>
        <adjacent_to_name>Ionization_Device</adjacent_to_name>
        <function>provide</function>
        <media>electric_charge</media>
        <caption>electric_charge</caption>
    </adjacent_to>
<status>GREEN</status>
</subsystem>
```

Figure 2: Electron_Emitter Prototype

The Electon_Emitter subsystem example used previously is expressed in the Averiti XML notation at Figure 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Edited by Averiti -->
<subsystem>
    <subsystem_name>Electron_Emitter01</subsystem_name>
    <instance_of>Spacecraft.Electrostatic_Thruster.Electron_Emitter</instance_of>
    <number_made_of>0</number_made_of>
    <number_part_of>1</number_part_of>
    <part_of>Electrostatic_Thruster01</part_of>
    <primary_function>emit_electrons</primary_function>
    <subsystem_notes>A device which ensures electron current discharged from vehicle to prevent charge build up</subsystem_notes>
    <number_attached_to>1</number_attached_to>
    <attached_to>Primary_Structure01</attached_to>
    <number_adjacent_from>1</number_adjacent_from>
    <adjacent_from>Power_Bus01</adjacent_from>
    <number_adjacent_to>1</number_adjacent_to>
    <adjacent_to>
        <adjacent_to_name>Ionization_Device01</adjacent_to_name>
        <function>provide</function>
        <media>electric_charge</media>
        <caption>electric_charge</caption>
    </adjacent_to>
<status>GREEN</status>
</subsystem>
```

Figure 3: Electron_Emitter Subsystem

XML Systems

Systems are captured in the Averiti notation language as XML documents. As with subsystems, use of this notation allows them to be easily machine and human readable and allows them to be easily shared and acted upon if required by other applications.

The schema, system.xsd, is used to capture systems. Using the Averiti tool subsystems are captured as a separate XML file per subsystem. The schema must therefore capture both the subsystem name and the subsystem filename. A system must comprise at least one subsystem and this is enforced by the schema. The System schema is presented at Figure 4.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="system">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="subsystem_definition" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="subsystem_name" type="xs:string"/>
            <xs:element name="subsystem_file_name" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Figure 4: system.xsd

The Ion_Thruster example used previously, which is comprised of five subsystems, is expressed in the Averiti XML notation in accordance with system.xsd is shown in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Edited by Averiti -->
<subsystem_definition>
    <subsystem_name>Propellant_Tank01</subsystem_name>
    <subsystem_file_name>Propellant_Tank01.xml</subsystem_file_name>
</subsystem_definition>
<subsystem_definition>
    <subsystem_name>Feed_Mechanism01</subsystem_name>
    <subsystem_file_name>Feed_Mechanism01.xml</subsystem_file_name>
</subsystem_definition>
<subsystem_definition>
    <subsystem_name>Ionization_Device01</subsystem_name>
    <subsystem_file_name>Ionization_Device01.xml</subsystem_file_name>
</subsystem_definition>
<subsystem_definition>
    <subsystem_name>Electrostatic_Accelerator01</subsystem_name>
    <subsystem_file_name>Electrostatic_Accelerator01.xml</subsystem_file_name>
</subsystem_definition>
<subsystem_definition>
    <subsystem_name>Electron_Emitter01</subsystem_name>
    <subsystem_file_name>Electron_Emitter01.xml</subsystem_file_name>
</subsystem_definition>
</system>
```

Figure 5: Ion_Thruster System