servX0: Semi-automatic services radiation length calculation.

ATLAS Project Doc. No: **None**	*Institute Doc. No:* **None**	*Created:* **1/3/2012** *Modified:* **1/3/2012**	*Page:* **1 of 9** *Rev:* **C**

Note

servX0: Semi-automatic services radiation length calculation.

The program servX0 has been written to help estimate radiation lengths of services for the new ATLAS inner-tracker at the HL-LHC. This document describes the program and gives some pointers on how to use it.

Prepared by: **N.P. Hessey**	*Checked by:* **Nobody**	*Approved by:* **Nobody**

Distribution List:
Public

History of Changes

Rev. No.	*Date*	*Pages*	*Made by*	*Description of changes*
A	1/3/2012	All	N.P. Hessey	Original document
B	29/5/2012	Many	N.P. Hessey	Add cross-sectional areas and .csv output
C	6/2/2014	9	N.P. Hessey	Corrected eq. 9

Contents

1 Introduction **3**

2 Installation **3**

3 DTD file **4**

4 XML Services description file **5**

4.1 Materials: `<material>` 6

4.2 Service volumes: `<serviceVolume>` 6

4.3 Service items: `<component>`, `<sheet>`, `<connector>` 6

4.4 Detector staves/petals/discs: `<multiModule>` 7

4.5 Detector layers: `<sensitiveLayer>` 7

5 Running servX0 **7**

6 Output files **7**

6.1 Main output 7

6.2 Output for spreadsheet processing 8

6.3 Output for idres 8

7 Calculations inside servX0 **8**

7.1 Component contribution to radiation length 8

1 Introduction

Traditionally services estimates for ATLAS tracker designs have been made in large spread-sheets. These can become very cumbersome and difficult to maintain; it is also hard to spot errors. Another approach to data handling is to use xml, which allows one to build up a hierarchy of tags to describe a collection of data. A document type definition (DTD) file gives the rules for the hierarchy. Standard tools then allow easy checking of a given xml file for consistency with the DTD. Many languages have interfaces to xml; servX0 is written in C++ using the libxml++ library. This allows the whole tree to be verified and read in to an internal data structure with essentially one line of code. It provides ways of traversing the data structure searching for different tags and values.

The user builds up an xml file describing a detector design. This must follow the DTD file rules. servX0 reads the xml file, validates it against the DTD file, and then processes the tree, building up the material contents of user-defined service volumes. It calculates the radiation length, mass, and cross-sectional area of services in each volume and prints these out in an ascii text file. It also produces a summary file of the services volumes in a format suitable for input to the program idres [1], and a .csv file suitable for input to spreadsheet programs.

2 Installation

Needs libxml++ installed on your system.

A tar-file is available at `http://www.nikhef.nl/~r29/software/`.

If installing from the tarfile, unpack the tarfile with something like:

```
> mkdir servX0 servX0/trunk
> mv servX0.tgz servX0/trunk
> cd servx0/trunk
> tar xzf servX0.tgz
```

Alternatively, use SVN to obtain the latest version:

```
> mkdir ~/svn
> cd ~/svn
> svn checkout svn+ssh://svn.cern.ch/reps/atlasusr/hessey/servX0
```

Build:

```
> cd servX0/trunk/src
> vi makefile
...edit the makefile to suit your set-up. Mainly set the BIN variable. Set
the version number in libxml++ stuff to the version installed on your
computer.
> make clean; make; make install
```

3 DTD file

The DTD file is available at `http://www.nikhef.nl/~r29/servX0.dtd`.

It defines the tags and their hierarchy. The xml file has to follow this definition file. It is reproduced below, as a guide to make sense of some of the input description.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ENTITY  PI "3.1415927">

<!ELEMENT detector (material+, serviceVolume+, tubeScaleFactor?,
                    (component|sheet|connector)+,
                    multiModule+, sensitiveLayer+)>

  <!ATTLIST detector name CDATA "Unnamed detector">

  <!ELEMENT material (density, X0)>
    <!ATTLIST material name ID #REQUIRED>
    <!ATTLIST material source CDATA #IMPLIED>

    <!ELEMENT density (#PCDATA)>
    <!ELEMENT X0 (#PCDATA)>

  <!ELEMENT serviceVolume (r1, z1, r2, z2)>
    <!ATTLIST serviceVolume name ID #REQUIRED>

    <!ELEMENT r1 (#PCDATA)>
    <!ELEMENT z1 (#PCDATA)>
    <!ELEMENT r2 (#PCDATA)>
    <!ELEMENT z2 (#PCDATA)>

  <!-- Components have the same length as the service volumes they
       cross ... and scale with the number of staves
       The scale factor is to account for wiggles, clips etc. -->
  <!ELEMENT tubeScaleFactor EMPTY>
    <!ATTLIST tubeScaleFactor value CDATA #REQUIRED>

  <!ELEMENT component (tube+)>
    <!ATTLIST component name ID #REQUIRED>

    <!ELEMENT tube (ri, ro)>
    <!ATTLIST tube material IDREF #REQUIRED>

      <!ELEMENT ri (#PCDATA)>
      <!ELEMENT ro (#PCDATA)>
```

```
<!-- Sheets are constant thickness throughout the service areas they
     cross...and are independent of the number of staves -->
<!ELEMENT sheet (thickness)>
  <!ATTLIST sheet name ID #REQUIRED>
  <!ATTLIST sheet material IDREF #REQUIRED>
  <!ATTLIST sheet serviceRoute IDREFS #REQUIRED>

  <!ELEMENT thickness (#PCDATA)>

<!-- Connectors have fixed size independent of the service area
     they are in ... and scale with the number of staves -->
<!ELEMENT connector (materialMass+)>
  <!ATTLIST connector name ID #REQUIRED>

  <!ELEMENT materialMass (#PCDATA)>
  <!ATTLIST materialMass material IDREF #REQUIRED>

<!ELEMENT multiModule (has+)>   <!-- e.g. stripStave, pixelStave-->
<!ATTLIST multiModule name ID #REQUIRED>

  <!ELEMENT has EMPTY>
  <!ATTLIST has quantity  CDATA #REQUIRED>
  <!ATTLIST has component IDREF #REQUIRED>


<!-- Number of staves/petals per barrel/disc -->
<!ELEMENT sensitiveLayer (quantity, connectorsVolume*)>
  <!ATTLIST sensitiveLayer name ID #REQUIRED>
  <!ATTLIST sensitiveLayer multiModule IDREF #REQUIRED>
  <!ATTLIST sensitiveLayer serviceRoute IDREFS #REQUIRED>

  <!ELEMENT quantity (#PCDATA)>

  <!ELEMENT connectorsVolume EMPTY>
    <!ATTLIST connectorsVolume connectors IDREFS #REQUIRED>
    <!ATTLIST connectorsVolume volume IDREF #REQUIRED>
```

4 XML Services description file

See the DTD file described above for full rules for the xml file. We give here a brief introduction to the intention of each tag. No attempt is made to give a complete description of how to define a services layout; look at the itkmedium.xml file in the examples directory and work from there, using the description given below to help understand the input format.

Note that an xml file can be spread over several files, with one main file inserting the other files as directed. This allows considerable re-use of information: you can keep one "materials" file and use it for many layouts. Or a basic list of components in another file. And so on.

The entire description is given inside a detector element, which just gives it a name:

```
<detector name="ATLAS ITk">
...
</detector>
```

4.1 Materials: `<material>`

The user must specify a set of materials which have a name, a density, and a radiation length (in g/mm^3 and g/mm^2, note the use of mm throughout while tables in e.g. the PDG booklet use cm).

A scale factor can be given. This is applied to all tube-type services (and nothing else). It is there to allow for the fact that tubes do not necessarily go in straight lines, and need holders etc. to keep them in place.

4.2 Service volumes: `<serviceVolume>`

A list of serviceVolumes is given; these are the main goal of the program – to calculate how much material is in each. Each is given a name and location; the location is given as (r,z) coordinates of the two endpoints.

The name is used later, for example in serviceRoute descriptions for sensitiveLayers.

4.3 Service items: `<component>, <sheet>, <connector>`

Then the items making up the services are given:

component: a better name might have been interlinks. These represent things like cables, tubes, and fibres which increase in mass in proportion to the length of a service volume. They are typically routed through many service volumes. For a given layout they scale with the number of staves using them.

sheet: These are things like a carbon-fibre disc, aluminised-kapton vapour barrier, or mimicking a support structure (which in reality is localised/lumpy) by its average thickness. Sheets have a fixed radiation length independent of a service volume area. They are typically routed through a few service volumes, which are listed here as the serviceRoute item. Sheets are not linked to the number of staves etc. in a layout.

connector: Things like a VCR connector, fibre-optic plug etc. They have a mass and material-type. The mass does not scale with service-volume area; the radiation length contribution is inversely proportional to the service-volume area. They are in a specific single service volume. They scale with the number of staves etc. in a layout.

Connectors once defined can be used in any multiModule using the <has> tag, giving the quantity and component attributes. The connector is put in a single service volume, using the <connectorsVolume> tag inside a <sensitiveLayer> tag. The connectors and volume attributes define which connector goes in what service volume.

4.4 Detector staves/petals/discs: <multiModule>

Items such as a short-strip stave, an end-cap strip petal, etc. are collectively known as multiModules. Each multiModule is given a name, as well as how many service components are attached to it.

4.5 Detector layers: <sensitiveLayer>

The multiModules are built up into sensitiveLayers. E.g. a barrel short strip layer could have 28 short-strip stave multiModules. The routing for the component services from a sensitiveLayer is defined as a list of serviceVolumes.

Also any connectors used by the sensitiveLayer are attached to the layer, and given a (single) service volume to be placed in.

5 Running servX0

```
> servX0 <xml-file>
```

where <xml-file> is a file containing valid xml describing a set of services. The file is processed and output directed to standard out. You probably want to redirect this to a file for further study.

6 Output files

6.1 Main output

The main output is plane text directed to standard out. In the examples directory there is a file itkmedium.out with an example of the output.

Begin at the end, in the Radiation Lengths summary table: You see a list of service volumes, with names like ServStripB3B4: this is a service-volume in the strip region, going from barrel 3 to 4. Each has a width (2π x mean-r) and a length (distance from B3 to B4).

Then for each it gives a list of what contributes material to this volume: the strip barrel support, pixel staves and discs, short strips and long strips all contribute. It gives the totals of how many of each item or stave-type contributes to the volume. Each item or stave-type contributes a mass in g and a radiation-length in %X0 for perpendicular-crossing tracks. You can see which services dominate, and which are less important, and you can see the totals for this volume. The cross-sectional areas of all tubular services are also given, to help estimate the space needed for services.

Above this section, in the Detector Layer Summary, is a summary of the layout: how many staves of each type are in each sensitiveLayer.

If you scroll up further in the file, Stave Types Summary, you find a break-down of the services of each multiModule. So for example a ShortStripStave has a cooling inlet and outlet, 2 LV cable pairs suitable for SS staves (LVSS), HV cables, some fibres and some DCS wires.

You see the contribution of each at the right-hand side: this is in strange units of "%X0 times width"; later this is divided by the width of the services region to get %X0. While an unusual unit (explained in section 7), the ratio of these numbers gives the relative contribution of each service-type for that multiModule. The cross-sectional area of each type of service, and the total area, are also given.

Looking further up the file, Connectors, Components, and Sheets Summary Tables, you can find the definition of each service. For example, the HV cables are coaxial cables, 2 x 0.83 mm diameter inner lead, 0.6 mm diam. insulator, 0.16 mm thick return/shield (which is excessive since the shield is meshed copper). The cross-sectional area of each component is estimated. The assumption is that the area is πr_{max}^2 where r_{max} is the radius of the biggest tube in the component. This of course can be wrong, e.g. for oval cables or if the user gives many cables of the same size, but not in an outer sheath. So beware: if you want to use this information, make sure you have such a sheath, even if it has an irrelevant wall thickness to avoid adding mass or radiation length.

At the top of the output, Materials Summary Table, is a table of the materials used.

6.2 Output for spreadsheet processing

The same information as in the main output is also printed as a .csv (comma separated values) file. This can easily be read in with most spreadsheet programs, allowing further processing.

6.3 Output for idres

In the input xml, serviceVolumes are also given their geographic location. This is not needed for the radiation length calculation, but does allow servX0 to give service volume descriptions in the format needed for the idres program. This output is written to the file idres.geom. It can be concatenated to an idres geometry file containing the active detector layers and run with idres to obtain a plot of %X0 vs. eta (as well as the track resolutions for which idres is intended).

7 Calculations inside servX0

7.1 Component contribution to radiation length

Define x_i as the fractional radiation length of a material in a component labelled by i, in a particular service volume. The contribution to the usual measure of radiation length is then

$$\%X0_i = 100x_i \tag{1}$$

with

$$x_i = \frac{m_i}{AX0_i} \tag{2}$$

where m_i is the mass of the material of that component, A is the area of the service volume, and $X0_i$ is the radiation length of the material (in mass per unit area, g/mm^2). The mass is

$$m_i = \rho_i a_i l \tag{3}$$

where ρ_i is the material density, a_i is its cross-sectional area, and l is the length of the services area which has to be traversed by the components in it. We write the area as

$$A = wl \tag{4}$$

with w the width of the service volume. For a cylindrical service volume,

$$A = 2\pi rl = wl \tag{5}$$

with

$$w = 2\pi r \tag{6}$$

For an annular service volume with inner and outer radii r_i and r_o,

$$A = \pi(r_o^2 - r_i^2) = \pi(r_o + r_i)(r_o - r_i) = wl \tag{7}$$

with

$$w = \pi(r_o + r_i) \tag{8}$$

Then for both cylindrical and annular service volumes,

$$wx_i = \rho_i a_i / X0_i \tag{9}$$

The RHS is independent of the geometry of the services volume; it depends only on the component and material properties. It is calculated for each component and stored. This is the quantity printed (as per cent) in the Component Summary Table. Later, these contributions are summed over all components in the volume and divided by the volume width to obtain the radiation length:

$$\%X0 = \frac{100}{w} \sum_i \frac{\rho_i a_i}{X0_i} \tag{10}$$

Similarly the total mass m in a service volume is found from

$$m = l \sum_i \rho_i a_i \tag{11}$$

where the sum is over all materials of all components. $\rho_i a_i$ is the mass per unit length of a material of a component. This is also printed for each component and totalled for each service volume.

References

[1] idres, a program for calculating track error matrices. N.P. Hessey, `http://www.nikhef.nl/~r29/upgrade/idres.html`