SANDIA REPORT

SAND2017-8890
Unlimited Release
Printed August 2017

Recommendations on a Document Structure Format for Automatic Report Generation

Aidan Hendrickson, Philippe P. Pébaÿ

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

SAND2017-8890
Unlimited Release
Printed August 2017

Recommendations on a Document Structure Format for Automatic Report Generation

Aidan Hendrickson, Philippe P. Pébaÿ
08753, 08753
Sandia National Laboratories
Livermore, CA 94551
U.S.A.

Abstract

In this report we propose a new, extensible and flexible methodology to describe the structure of documents for the Automatic Report Generator (ARG) currently being developed at Sandia.

Acknowledgments

The authors of this report wish to express their gratitude to Robert Clay and Nathan Spencer (Sandia) for supporting this work, and to thank Ernest Friedman-Hill (Sandia) for offering advice as to how the current work might be interfaced with the Next Generation Workflow.

They also acknowledge that the example pictures used to illustrate this document were generated from an exemplar data set created by Nathan Spencer, and allowed for unlimited release with review and approval number SAND2017-5827.

Contents

List of Figures

List of Tables

This page intentionally left blank.

Chapter 1

Introduction

1.1 Disclaimer

This report was generated using a new data input structure for the Automatic Report Generator (ARG) currently being developed at Sandia. However, as the ARG is still work in progress; the present document should be viewed as a proof of concept report.

Furthermore, this document is about a proposed interface between the ARG and other tools; e.g., a graphical user interface allowing a user to visually assemble a document structure. The schemata described below are therefore meant for developers of such applications, not for prospective users of the ARG.

1.2 Background

Previous versions of the ARG have accepted structural input in the form of a comma-separated values (CSV) file format. However, this *ad hoc* format, introduced in the early phase of ARG development in order to allow for immediate production of example documents, is inflexible and hardly legible to a novice human user, particularly for complex report artifacts such as visualizations.

Meanwhile, this CSV format has the advantage of being easily interfaced with the Sandia Analysis Workflow (SAW) toolchain. While moving away from CSV will likely make things easier for both users and developers of the ARG, compatibility with SAW may be an important factor in deciding which alternative data format is best.

1.3 JSON and YAML

JSON can be seamlessly integrated into the dictionary and list structures of JavaScript and Python; for this reason in particular, it is widely used for web applications. It is more efficient than XML. However, the dictionary-like syntax requires large numbers of carefully placed commas and quotation marks, making it difficult to write and edit. Furthermore, JSON does not support comments.

YAML is a recursive acronym that stands for "YAML Ain't Markup Language". Being a superset of JSON, it supports most JSON syntax as well as several new features including comments. Designed for human input rather than data exchange, YAML syntax is simple and easy to use, organized by indentations instead of brackets. There are two libraries that support YAML in python. PyYAML is no longer maintained, so the ARG implements ruamel.yaml. The library can be installed with he following command:

```
pip install ruamel.yaml.
```

In contrast, JSON is slightly more compact and efficient to parse and requires no dependency. While YAML is becoming more widespread, so is JSON (as XML declines), which is currently the dominant data interchange format on the web.

1.4 Other Data Formats

XML is a widely used text markup language, designed to process data that is primarily text—even so, it has been widely used for data exchange on the web. While it remains popular, its use is now steeply declining—According to Google Trends, searches for JSON first surpassed those for XML in 2016.

XML handles text well—an advantage would allow the various supporting text files to be combined into a single file. However, data for charts and images would be messy. Embedded JSON could solve this problem, but would reduce simplicity and portability.

XML is less efficient (in terms of space and time), less readable, and declining in popularity when compared to JSON and YAML, thus XML routes appears to be riskier in terms of both ease of use and long-term maintainability.

Chapter 2

Recommendations

2.1 A New YAML-Based Schema

This SAND report was created by the Automatic Report Generator (ARG) with the following command was issued:

```
python ARG.py -u -s s.yaml -r SAND -c constants.in -a abstract.txt -t thanks.txt
```

where `s.yaml` is a YAML data file describing the high-level structure of the report; this report was generated with the following structure:

```
#Title directive
title:
  type: sd
  file: Modal.inp
#Report
chapters:
  - n: tex_chapter
    title: Introduction
    sections:
    - n: tex_section
      title: Disclaimer
      text: disclaimer.tex
    - n: tex_section
      title: Background
      text: history.tex
    - n: tex_section
      title: JSON and YAML
      text: yaml_paragraph.tex
    - n: tex_section
      title: Other Data Formats
      text: other_formats.tex
  - n: tex_chapter
    title: Recommendations
```

```
sections:
- n: tex_section
  title: A New YAML-Based Schema
  sections:
    - n: tex_paragraph
      text: introduction.tex
    - n: tex_verbatim
      text: SAND2017-XXXX.yaml
- n: tex_section
  title: A New JSON-Based Schema
  sections:
    - n: tex_paragraph
      text: data_formats.tex
    - n: tex_verbatim
      text: SAND2017-XXXX_auto.json
    - n: tex_paragraph
      text: recursion.tex
- n: tex_section
  title: Examples
  text: vtk_demonstration.tex
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: surface
  opacity: .5
- n: vtk
  width: 8cm
  view_direction: [11.1, -14.8, 0.0]
  model: Package.e
  var_name: ObjectId
  render: surface_with_edges
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: clip
  normal_vector: [0.,1.,.8]
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: clip
  normal_vector: [0.,1.,.8]
  #optional parameters:
  opacity: .5
  ghost_opacity: .1
  ghost_wireframe: True
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: multiclip
  normal_vectors: [[0,1,1],[0,1,0],[-1,0,.5]]
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: multiclip
  normal_vectors: [[0,-1,1],[0,1,1],[-1,0,0]]
  #optional parameters:
  ghost_opacity: 1
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: multicut
  #optional parameters:
  slice_number: 9
- n: vtk
  width: 8cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
  var_name: ObjectId
  render: 3Dcut
  normal_vector: [0,2,1]
  #optional parameters:
  slice_number: 1
  ghost_opacity: .05
  ghost_wireframe: True
- n: vtk
  width: 16cm
  view_direction: [15.0, -35.0, 0.0]
  model: Package.e
```

```
      var_name: ObjectId
      render: slice
      normal_vector: [0,1,1]
      #optional parameters:
      slice_number: 20
      slice_width: 1
      gap_width: 1
  - n: tex_chapter
    title: Conclusion
    text: recommendations.tex
```

2.2 A New JSON-Based Schema

What follows is the same report structured, converted programmatically into JSON—this JSON file could also be used as input to generate this report. With either input file type, reading the following as a series of nested dictionaries and lists in python, this was the data structure used by the ARG to generate this report:

```
{
  "chapters": [
    {
      "n": "tex_chapter",
      "sections": [
        {
          "text": "disclaimer.tex",
          "n": "tex_section",
          "title": "Disclaimer"
        },
        {
          "text": "history.tex",
          "n": "tex_section",
          "title": "Background"
        },
        {
          "text": "yaml_paragraph.tex",
          "n": "tex_section",
          "title": "JSON and YAML"
        },
        {
          "text": "other_formats.tex",
          "n": "tex_section",
          "title": "Other Data Formats"
        }
      ],
      "title": "Introduction"
    },
    {
      "n": "tex_chapter",
      "sections": [
        {
          "n": "tex_section",
          "sections": [
            {
              "text": "introduction.tex",
              "n": "tex_paragraph"
            },
            {
              "text": "SAND2017-XXXX.yaml",
              "n": "tex_verbatim"
            }
          ],
          "title": "A New YAML-Based Schema"
        },
        {
          "n": "tex_section",
          "sections": [
            {
              "text": "data_formats.tex",
              "n": "tex_paragraph"
            },
            {
              "text": "SAND2017-XXXX_auto.json",
              "n": "tex_verbatim"
            },
            {
              "text": "recursion.tex",
              "n": "tex_paragraph"
            }
          ],
          "title": "A New JSON-Based Schema"
        },
        {
          "text": "vtk_demonstration.tex",
          "n": "tex_section",
          "title": "Examples"
        },
        {
```

```
    "opacity": 0.5,
    "width": "8cm",
    "var_name": "ObjectId",
    "render": "surface",
    "model": "Package.e",
    "view_direction": [
      15.0,
      -35.0,
      0.0
    ],
    "n": "vtk"
  },
  {
    "width": "8cm",
    "var_name": "ObjectId",
    "render": "surface_with_edges",
    "model": "Package.e",
    "view_direction": [
      11.1,
      -14.8,
      0.0
    ],
    "n": "vtk"
  },
  {
    "width": "8cm",
    "var_name": "ObjectId",
    "render": "clip",
    "model": "Package.e",
    "view_direction": [
      15.0,
      -35.0,
      0.0
    ],
    "normal_vector": [
      0.0,
      1.0,
      0.8
    ],
    "n": "vtk"
  },
  {
    "opacity": 0.5,
    "render": "clip",
    "view_direction": [
        15.0,
        -35.0,
        0.0
      ],
      "normal_vector": [
        0.0,
        1.0,
        0.8
      ],
      "n": "vtk",
      "width": "8cm",
      "var_name": "ObjectId",
      "ghost_opacity": 0.1,
      "model": "Package.e",
      "ghost_wireframe": true
    },
    {
      "width": "8cm",
      "var_name": "ObjectId",
      "render": "multiclip",
      "model": "Package.e",
      "view_direction": [
        15.0,
        -35.0,
        0.0
      ],
      "normal_vectors": [
        [
          0,
          1,
          1
        ],
        [
          0,
          1,
          0
        ],
        [
          -1,
          0,
          0.5
        ]
      ],
      "n": "vtk"
    },
{
  "render": "multiclip",
  "view_direction": [
    15.0,
    -35.0,
    0.0
  ],
  "n": "vtk",
  "width": "8cm",
  "var_name": "ObjectId",
  "model": "Package.e",
  "ghost_opacity": 1,
  "normal_vectors": [
    [
      0,
      -1,
      1
    ],
    [
      0,
      1,
      1
    ],
    [
      -1,
      0,
      0
    ]
  ]
},
{
  "width": "8cm",
  "var_name": "ObjectId",
  "render": "multicut",
  "model": "Package.e",
  "view_direction": [
    15.0,
    -35.0,
    0.0
  ],
  "slice_number": 9,
  "n": "vtk"
},
{
  "render": "3Dcut",
          "view_direction": [
            15.0,
            -35.0,
            0.0
          ],
          "normal_vector": [
            0,
            2,
            1
          ],
          "n": "vtk",
          "width": "8cm",
          "var_name": "ObjectId",
          "ghost_opacity": 0.05,
          "model": "Package.e",
          "ghost_wireframe": true,
          "slice_number": 1
        },
        {
          "gap_width": 1,
          "render": "slice",
          "slice_width": 1,
          "view_direction": [
            15.0,
            -35.0,
            0.0
          ],
          "normal_vector": [
            0,
            1,
            1
          ],
          "n": "vtk",
          "width": "16cm",
          "var_name": "ObjectId",
          "model": "Package.e",
          "slice_number": 20
        }
      ],
      "title": "Recommendations"
    },
    {
      "text": "recommendations.tex",
      "n": "tex_chapter",
      "title": "Conclusion"
    }
  ],
  "title": {
    "type": "sd",
    "file": "Modal.inp"
  }
}
```

In contrast to CSV, this data format supports recursion: i.e. chapters and sections can be written in a nested format (as opposed to simply marking the start of each new section). As a result, recursive structures or formatting tools could be added to the ARG.

More importantly, the nested structure makes it intuitive to follow the natural hierarchy of chapters, sections, and further subdivisions: the input now resembles an outline.

Furthermore, this nested structure is flexible and optional, and it currently has no effect on the appearance of the report.

2.3 Examples

The following illustrations demonstrate the capabilities of the new data format for handling images. This data format allows for optional parameters that default to a value when not provided in the input (i.e. opacity defaults to 1). This feature can be added to more parameters in the future including size, view angle, and normal angle.

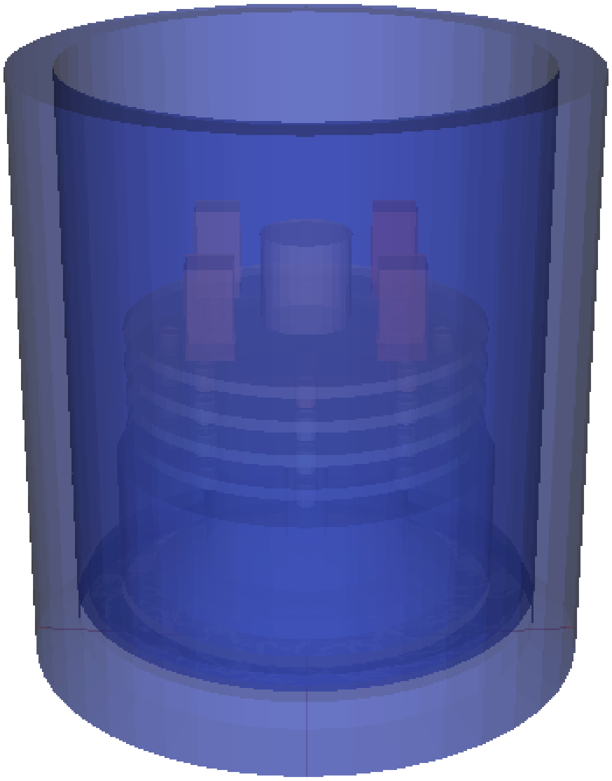

Figure 2.1. Translucent surface rendering of *Package.e* colored by *ObjectId*.

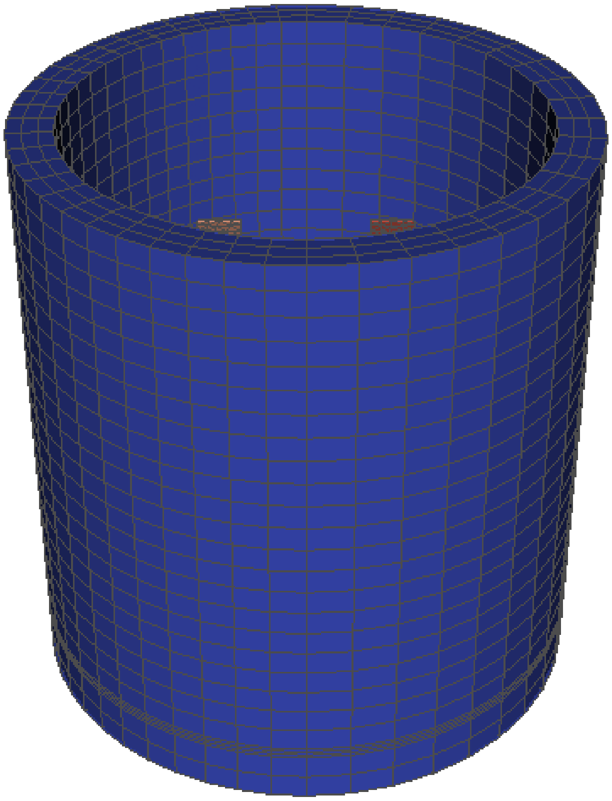

Figure 2.2. Surface rendering of *Package.e* colored by *ObjectId*.

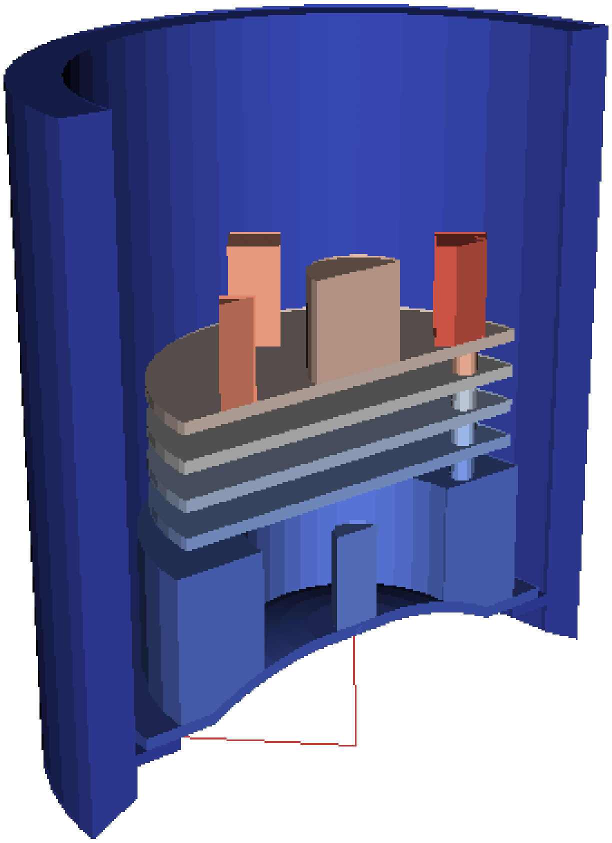

Figure 2.3. Plane clip of *Package.e* colored by *ObjectId*.

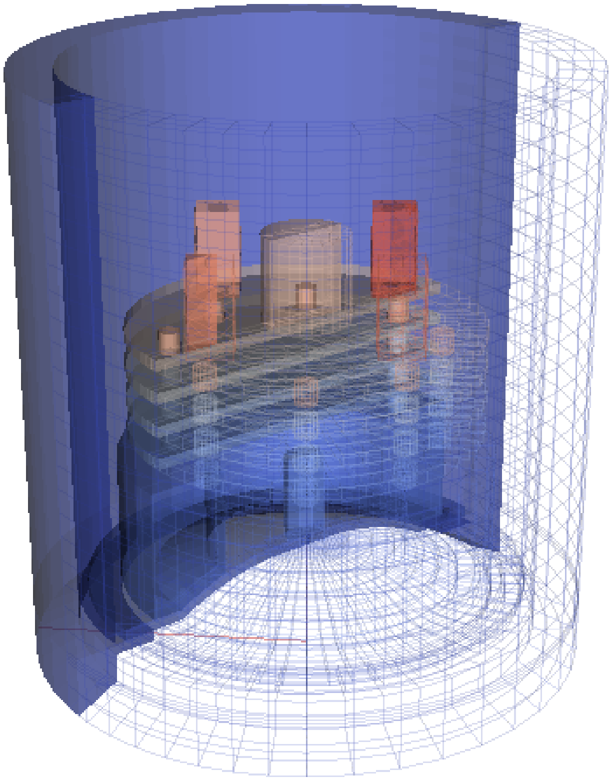

Figure 2.4. Plane clip of *Package.e* colored by *ObjectId*.

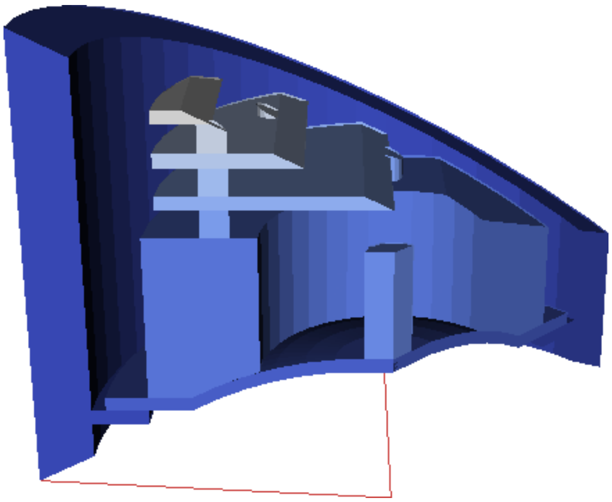

Figure 2.5. Plane multi-clip of *Package.e* colored by *ObjectId*.

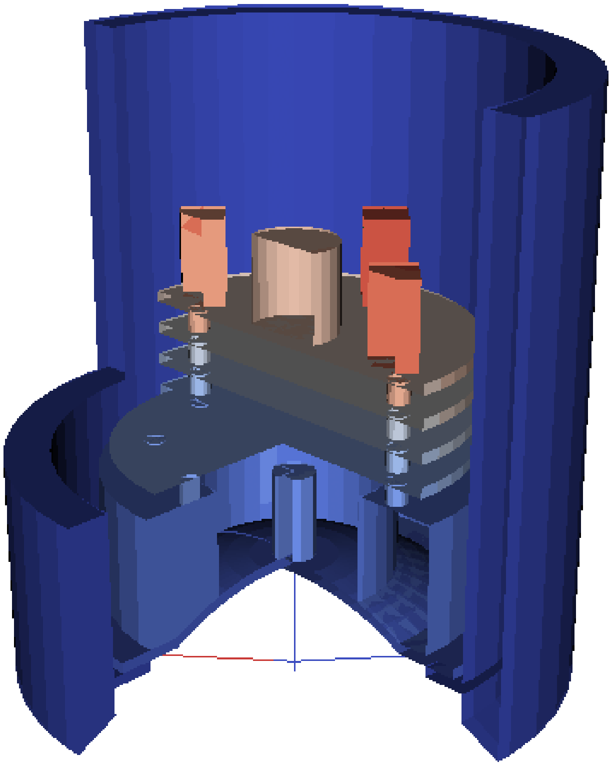

Figure 2.6. Plane multi-clip of *Package.e* colored by *ObjectId*.

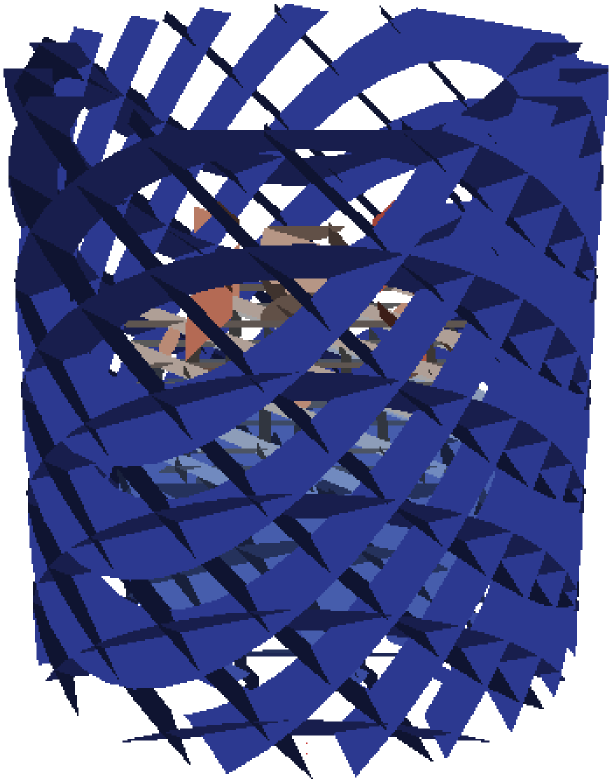

Figure 2.7. Perpendicular plane cuts of *Package.e* colored by *ObjectId*.

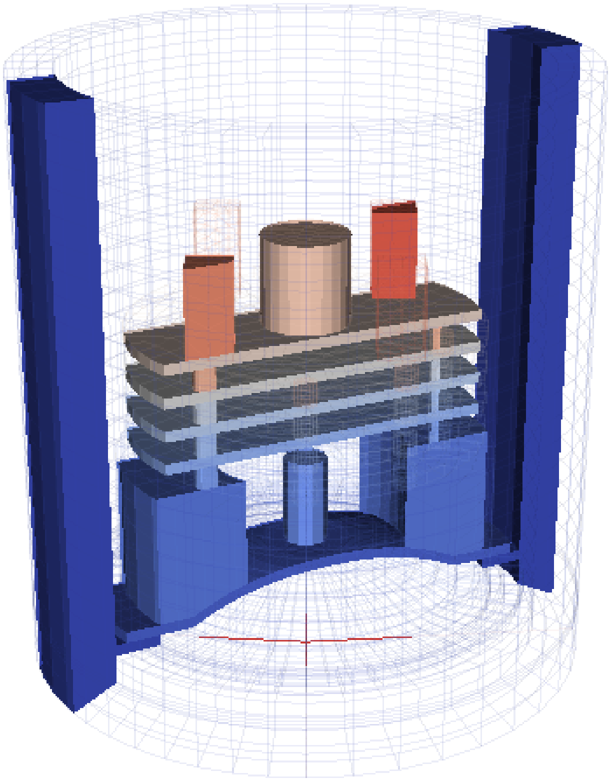

Figure 2.8. Sections of *Package.e* colored by *ObjectId*.

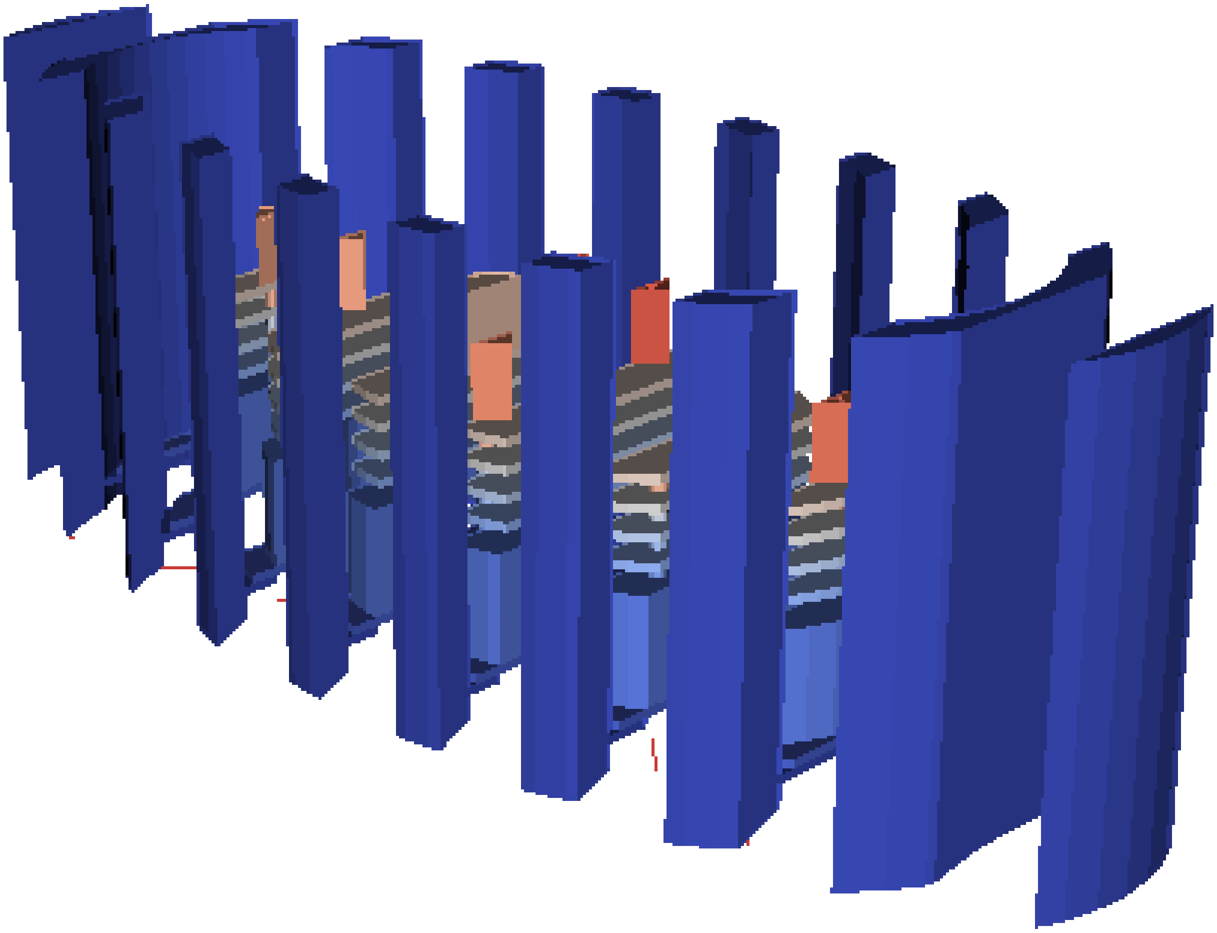

Figure 2.9. Spread cross-section of *Package.e* colored by *ObjectId*.

Chapter 3

Conclusion

The ARG currently supports JSON, YAML, and CSV.

For the transmission of data, JSON is the superior choice, thanks to its widespread use in many applications and libraries. YAML, however, is much friendlier to a user or developer interacting directly with this data.

We contend that, as long as the ARG depends on human-written data files, YAML is the appropriate choice. As development continues, support for one or more of these data types may be phased out as it is no longer useful.

Finally, we recommend that XML not be considered as a document structure description modality for automatic report generation.

DISTRIBUTION:

1	MS	9159	Aidan Hendrickson, 08753
1	MS	N/A	Philippe P. Pébaÿ, 08753
1	MS	0899	Technical Library, 8944 (electronic copy)

Sandia National Laboratories