Using the IBM® Rational® Publishing Engine to Streamline the Documentation Generation Process at General Dynamics Land Systems

Michael Sutherland
General Dynamics Land Systems
sutherla@gdls.com

IBM Software
Innovate2011
The Premier Event for Software and Systems Innovation

Software. Everywhere. June 5–9 Orlando, Florida

RDM-1272
Objective – Rational Publishing Engine (RPE)

- Provide a roadmap that allows someone new to RPE to learn how to create RPE templates or modify existing RPE templates

- Demonstrate the value of investing in RPE and replacing existing manual publishing processes
Publishing Process to Replace

- We use an in-house modified version of the standard DOORS “Export to Word”

- The most complicated documents are produced in three separate passes, and “hand-jammed” together
  - Main Document
  - Verification Cross-Reference Index (VCRI)
  - Verification Plan
Opportunity to Streamline

- Documents can take up to 30 minutes each to export, hand assemble, and hand finish the title page, headers, and footers.

- Some project milestones require dozens of documents to be produced by the same date.
  - Additional days are added to the schedule for document production, so the engineers have to put “pencils down” earlier than they would like.
IBM Rational Publishing Engine (RPE)

- Ease-of use
- One click regeneration
- Predefined templates
- Intuitively designed
- Speed, agility, and scaleability
- Deliver documents on time

How do we get started?

The Right Information at the Right Time

IBM® Rational® Publishing Engine is designed for **ease-of-use**. You don’t need to change the way you work to enjoy the benefits of Rational Publishing Engine. Built-in capabilities to extract data from a range of data sources can help reduce manual work and risk of errors. You have the option of generating documents within your preferred application such as IBM® Rational® DOORS® for requirements management, or IBM® Rational® Tau for modeling, or directly from the Rational Engine interface. Regeneration of a document including the latest data from the source applications, is **only a click away**.

Rational Engine includes **predefined templates** with guidelines that help you get up and running quickly. You can design the content and format of your documents with a full suite of tools for creating and editing templates **intuitively designed** with features such as drag-and-drop and preview capabilities. These capabilities, and more, can help you create documents quickly and easily.

**High performance document generation**

Rational Publishing Engine is designed for **speed, agility and scalability**.

The in-built capability to extract data from IBM Rational solutions and third-party products help you generate systems and software documentation in a timely manner. Capabilities such as **parallel processing** enhance your document generation performance, helping you **deliver documents on time**.

**High quality, elegant documents**

With Rational Publishing Engine, you get a document generation application that can help you create elegant, easy-to-read, multi-source documents in formats that include Microsoft Word®, PDF, and HTML, and XML/FO, with a completely flexible document style.

Documents generated with Rational Publishing Engine support an extensive range of features such as richly formatted text, tables of contents and OLEs. Rational Publishing Engine will automatically extract data from a single source (e.g. one or more modules in a Rational DOORS® database) or combine data from multiple sources (e.g. Rational DOORS® modules and IBM Rational® Tau® models). The advanced capabilities of Rational Publishing Engine will help you produce easily updated, high-quality documents.
Templates and Knowledge Base

- IBM Rational provides example Templates and an online Knowledge Base

The templates and knowledge base are good examples of what is possible, but no template matched our publishing requirements.

Action: Open example templates, read manual, explore knowledge base.

Result: Some knowledge is gained, but we still don’t know how to create a template that matches our publishing requirements.
Open Example Template

- Start -> All Programs -> IBM Rational -> IBM Rational Publishing Engine Document Studio
  - This opens the RPE Document Studio

**RPE Document Studio**

The Document Studio component of RPE allows designing document templates.

- Minimize the Welcome Screen
- File -> Open Document Template…
- Browse to:
  - C:\Program Files\IBM\Rational\Publishing Engine\1.1.1.2\source\DOORS\examples
  - Open “Tutorial_DOORS.dta”
Boxes within boxes?  What do they all do?  What do the icons mean?
What is a RPE Template?

Document template

This is the blueprint for the document generation. A document template defines what data is to be extracted from the data source (queries) and how to format all the information (formatting information).

A template is built of static and dynamic content. The static content is defined by the data such as texts and images provided when the template is designed. The dynamic content is represented by data obtained from the data sources at document generation time.

Formatting information can be defined in the template, while for some data sources; formatting information embedded in the data can also be retained.
No, really, what is a RPE Template?

- After working with RPE Templates for a few days, I realized I had seen something similar back in college.
Structured Programming

- A Nassi–Shneiderman diagram (NSD) is a graphical design representation for structured computer programming.
  - Similar in purpose to a flowchart
  - Uses nested boxes to represent subproblems (Repetition, Selection, Sequence)

```dxl
// DXL

// Repetition == for loop
for (x=1; x <= 11; x+=2) {
    print x
}

// Selection == if then
int i = 2
if (i < 3) {
    i += 2
}

// Sequence == statement(s)
print x
i += 2
```
Repetition, Selection, and Sequence in DOORS DXL

- **Repetition – Loop through Objects in Module**
  ```
  Object o
  for o in current Module do {
    // Do something for each Object
  }
  ```

- **Selection – Decide what to do for each Object based on Attributes**
  ```
  Object o = current Object
  if ( o."Object Heading" "" != "" ) {
    // Do something for this Object
  }
  ```

- **Sequence – Get and set Object Attributes**
  ```
  Object o = current Object
  print o."Object Heading" ""
  o."Object Text" = "The system shall...."
  ```
Repetition, Selection, and Sequence in RPE

- RPE achieves repetition through *queries*
- RPE achieves selection through *conditions*
- RPE achieves sequence through *queries* (and *assignments*, more on that later…)

- Container, Paragraph and Text are *elements* of the template

- From IBM provided “Tutorial_DOORS.dta”
RPE Queries

A query defines what data is to be extracted from the data source. You can assign a query to any template element. Template elements having queries will be generated for each data element extracted from the data source.

- List (collection) *queries* appear at the *top* of the element box
  - List of items is iterated through item by item
- Attribute *queries* appear *inside* the element box
  - Single value is returned

**RPE List Query for Objects in Module**

**RPE Attribute Query**

**RPE List Query for Links in Object**

**RPE List Query for Object at the other end of the link**
(in this case, the list contains only one item, the other end of the source/target pair of Objects)
RPE Conditions

Conditions

Using expressions based on data attributes or template variables you can define conditions for when an element should be rendered. The condition is a JavaScript expression that evaluates to a **Boolean** value.

- Similar in concept to DOORS filtering
RPE – DOORS Native Filters vs. Conditions

**Filtering data**

There are situations when not all the data is needed. In these cases you can limit the amount of processed data by setting a filter on the query. There are two ways for specifying a filter:

- *RPE Filter* – JavaScript expression using the data attributes of the entities returned by the query
- *Native filter* – plain text in a format specific to each data source type.

When the query is performed, only the data entities matching the filter will be included in the output.

**Filter vs. Conditions**

While looking similar in purpose the Filters and Conditions are two different mechanisms that serve different purposes.

A filter can be evaluated as data is retrieved from the data source.

A condition is evaluated only after the data has been extracted from the data source. So while you can use conditions instead of filters, using filters yields better performance as RPE will only process a subset of the data.

**TIP** Native filters are usually more effective and yield faster document generation times than RPE filters. Whenever possible try using native filters.

- We have been using JavaScript Conditions, need to see if we can replace any Conditions with DOORS Native Filters and check for any performance improvements.
Set Condition using Script Expression in RPE

(1) Select element

(2) Access the Condition Property for the Text Element containing the Object Heading query
Set Condition using Script Expression in RPE (continued)

(3) Use dialog box to set condition

(4) Element with a condition will display the following symbol:
Tutorial of the Mechanics of Template Development

At this point, one should get comfortable with the mechanics.

http://www.ibm.com/developerworks/wikis/display/rpe/Simple+DOORS+Template

Pre-requisites:

- RPE 1.1
- Java 6 or later
- DOORS 9.1 or later
- a supported Windows version

Tutorial

Follow the attached MS PowerPoint Slideshow: link...
The example template is: link...

Action: Download and go through the tutorial.
After completing the tutorial, you will have learned how to create and execute the following simple RPE template:

**Conditions:**

- `Object_Heading != ""`
- `Object_Text != ""`

After executing the template, you will be asking yourself the following questions:
Why are the heading numbers not displayed?

**Numbering headings for Microsoft Word**

For headings to be numbered as a hierarchical list the easiest way is to use a stylesheet that has the headings numbered.

- **Problem:** We have many “Product Line” modules, in which not all headings will export for each version of the product due to Product Line filtering.
  - Following the “easy” way leads to the heading numbers generated by the stylesheet being different than the actual heading numbers in DOORS.

Why are the headings inconsistently styled?

- **Problem:** The default Heading styles in Word are inconsistent
  - Different bolding, italics, font family, font size for each Heading style
Exported Headings before modifying RPE Template

<table>
<thead>
<tr>
<th>Object Level</th>
<th>1 Level One Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Level Two Heading</td>
</tr>
<tr>
<td>2</td>
<td>Level Three Heading</td>
</tr>
<tr>
<td>3</td>
<td>Level Four Heading</td>
</tr>
<tr>
<td>4</td>
<td>Level Five Heading</td>
</tr>
<tr>
<td>5</td>
<td>Level Six Heading</td>
</tr>
<tr>
<td>6</td>
<td>Level Seven Heading</td>
</tr>
<tr>
<td>7</td>
<td>Level Eight Heading</td>
</tr>
<tr>
<td>8</td>
<td>Level Nine Heading</td>
</tr>
</tbody>
</table>

Level One Heading

Level Two Heading

Level Three Heading

Level Four Heading

Level Five Heading

Level Six Heading

Level Seven Heading

Level Eight Heading

Level Nine Heading
Displaying DOORS Heading Numbers

- Add styled Paragraph, query for “Heading Number” (DOORS “Object Number”), add whitespace, export “Object Heading” as script

Using object level here ensures the entire Paragraph is styled “Heading 1” thru “Heading 9”

(See “How to, Tips & tricks, Heading Styles” in RPE Manual)

Using a direct attribute query causes an unwanted line feed, must query by script instead

Add whitespace (two spaces or tab character)

Word will default to inconsistent styles for “Heading 1” thru “Heading 9”, use RPE to force consistency,
Exported Headings after modifying RPE template

<table>
<thead>
<tr>
<th>Object Level</th>
<th>1 Level One Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.1 Level Two Heading</td>
</tr>
<tr>
<td>3</td>
<td>1.1.1 Level Three Heading</td>
</tr>
<tr>
<td>4</td>
<td>1.1.1.1 Level Four Heading</td>
</tr>
<tr>
<td>5</td>
<td>1.1.1.1.1 Level Five Heading</td>
</tr>
<tr>
<td>6</td>
<td>1.1.1.1.1.1 Level Six Heading</td>
</tr>
<tr>
<td>7</td>
<td>1.1.1.1.1.1.1 Level Seven Heading</td>
</tr>
<tr>
<td>8</td>
<td>1.1.1.1.1.1.1.1 Level Eight Heading</td>
</tr>
<tr>
<td>9</td>
<td>1.1.1.1.1.1.1.1.1 Level Nine Heading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heading 1</th>
<th>1 Level One Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading 2</td>
<td></td>
</tr>
<tr>
<td>Heading 3</td>
<td>1.1 Level Two Heading</td>
</tr>
<tr>
<td>Heading 4</td>
<td>1.1.1 Level Three Heading</td>
</tr>
<tr>
<td>Heading 5</td>
<td>1.1.1.1 Level Four Heading</td>
</tr>
<tr>
<td>Heading 6</td>
<td>1.1.1.1.1 Level Five Heading</td>
</tr>
<tr>
<td>Heading 7</td>
<td>1.1.1.1.1.1 Level Six Heading</td>
</tr>
<tr>
<td>Heading 8</td>
<td>1.1.1.1.1.1.1 Level Seven Heading</td>
</tr>
<tr>
<td>Heading 9</td>
<td>1.1.1.1.1.1.1.1 Level Eight Heading</td>
</tr>
<tr>
<td></td>
<td>1.1.1.1.1.1.1.1.1 Level Nine Heading</td>
</tr>
</tbody>
</table>
Styles and Stylesheets

- In RPE, each Paragraph Element can be individually styled, or a RPE style element can be created and managed as a single data item and assigned to multiple Elements by name.

- Given that RPE has Style Elements, do I need a Word .dot(x) template?
Word Templates .dot(x) – RPE calls them Stylesheets

**Stylesheets**

The HTML and Word output formats allow specifying stylesheets to be used in the document generation. A stylesheet can be used to ensure certain styles are available in the generated document. A stylesheet can also be used to add content to the generated document: standard front pages with company logos, legal information etc.

**Using external styles**

RPE can use styles that are not defined in the document template as long as they are provided in the stylesheet. All you need to do is type the name of the style in the element’s property page as you would do with a RPE style. When the output is generated RPE will try to use that style if present in the stylesheet. If a style with that name is not found, no style will be applied to that element.

**Style precedence**

When the document is generated, if RPE finds a style with the same name as a style defined in the document template, the stylesheet style is used. This applies to Word output only.

**Style inheritance**

If a style is applied to a template element that also has values set for individual formatting properties, the element’s changes will override the style ones if such an overlap exists.
No, really, do I need a Word .dot(x) template?

- **Answer: Maybe?**
  1. IBM Rational provides Word macros for post processing
     - Update all the Word tables of contents (TOC, TOT, TOF) in the document
     - Updates all the Word fields in the document (Figure and Table Numbers, etc.)
     - Embed the content of the OLE include file in the output document
     - Checks all the include file fields and places their content inside the document
  2. Without using a Word template, RPE exports all RPE Styles to “Normal” regardless of the name of the style in RPE
     - RPE Style settings are applied, but the name remains “Normal” for each Paragraph
     - If a Word template is used, and the RPE Style name matches the Word style name, the Style is named in Word
No, really, do I need a Word .dot(x) template? Continued

3. RPE does not have the functionality to insert a Watermark (draft, etc.)

4. Programming the cover page, front matter, headers and footers in RPE allows for dynamic in-line population of content (Document Number, Name, Version, etc.) using DOORS Attribute queries (Module level Attributes) or RPE Variables
   - If you have existing cover page or front matter boilerplate, it may be easier to leave it in a Word template
     - There is no File -> Import -> Office -> Word in RPE

   ▪ For reasons (1,2,3), we do use a minimal Word template for most exports but we have the cover page, front matter, headers and footers in RPE
     ▪ Word Template has
       ▪ Macros
       ▪ Style names
       ▪ Watermark
DOORS Native Tables

*Tables*

You need to explicitly query for DOORS tables as they are not extracted automatically from DOORS. While manually adding the queries for extracting a table requires extra effort when building the template, this approach has the advantage of allowing fine-grained control over the formatting of the table.

<table>
<thead>
<tr>
<th>Table</th>
<th>DOORS 1</th>
<th>$2</th>
<th>Module.Object.Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>DOORS 1</td>
<td>$3</td>
<td>Module.Object.Table.Row</td>
</tr>
<tr>
<td>Cell</td>
<td>DOORS 1</td>
<td>$4</td>
<td>Module.Object.Table.Row.Object</td>
</tr>
</tbody>
</table>

- With this configuration of Elements, DOORS Native Tables are exported, but the formatting in DOORS is not preserved
Formatting DOORS Native Tables

- The RPE provided template “doorsData.dta” has a good example of how to format DOORS Native Tables

<table>
<thead>
<tr>
<th>Container</th>
<th>Table with DOORS properties preserved: cell width, cell alignment, border style, cells number within row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>DS1 $3 Module/Object/Table/Row</td>
</tr>
<tr>
<td>Cell</td>
<td>DS1 $4 Module/Object/Table/Row/Object</td>
</tr>
<tr>
<td>Text</td>
<td>Module/Object/Table/Row/Object/Object Text</td>
</tr>
</tbody>
</table>

- Formatting uses DOORS Table Advanced System Attributes

- May need to execute “DOORS Schema Discovery Wizard” to pull in the Table System Attributes
Formatting DOORS Native Tables – Cell Alignment

- DOORS allows the user to set Cell Alignment for each cell, use Script Expression to recreate the formatting when using RPE to export.
JavaScript

- Advanced RPE Template development requires some knowledge of JavaScript
  - Variables
  - Syntax
  - Control structures (if then else)
  - Regular Expressions
    - Script Expressions cannot directly query DOORS Multi-Valued Enumerated Attributes for individual values
  - Date Formatting
RPE Variables and Assignment

**Variables**

RPE template variables are available to be used as placeholders for data calculated at runtime (variable assignments) or provided in the document specification. Unlike data attributes you can also use variables in master pages.

**Variable assignments**

When designing a template it might be necessary to calculate values or make available data attributes in contexts where normally they would not be available. This can be achieved through the mechanism of variable assignment.

The variable assignment allows the template designer to change the value of a variable at any given time. The new value can be static data or data available in the current context.
RPE Variables and Assignment - Example

- Some tables are longer than a page, need to repeat header row across pages
  1. Create Variable “firstRow”
  2. Assign “firstRow = true” for Table Element
  3. Assign “firstRow = false” for Cell Element
  4. This causes firstRow to be true for the first row, and false for subsequent rows
  5. For the row, set “row repeat at page begining” = firstRow

- The following symbol (×) shows when an assignment is present on an element
Verification Cross-Reference Index (VCRI)

### MIL-STD-961D

- This standard establishes the formats, contents, and procedures for the preparation of performance specifications, detail specifications, program-unique specifications, and associated documents, prepared either by Government activities or under contract.
- **Requirements in “Section 3” of the outline**

### MIL-STD-961D NOTICE 1 - VCRI (Verification Cross-Reference Index)

- FIGURE A-3. Sample requirements/verification cross-reference matrix for detail specification

<table>
<thead>
<tr>
<th>REQUIREMENT / VERIFICATION CROSS-REFERENCE MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD OF VERIFICATION</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>N/A - NOT APPLICABLE</td>
</tr>
<tr>
<td>1 - ANALYSIS</td>
</tr>
<tr>
<td>2 - DEMONSTRATION</td>
</tr>
<tr>
<td>3 - EXAMINATION</td>
</tr>
<tr>
<td>4 - TEST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION 3 REQUIREMENT</th>
<th>VERIFICATION METHOD</th>
<th>VERIFICATION CLASS</th>
<th>SECTION 4 VERIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.3.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>4.3.2.1.1</td>
</tr>
<tr>
<td>3.1.3.2</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>4.3.2.2.3</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>4.3.2.5.1</td>
</tr>
</tbody>
</table>

Creation of the VCRI with RPE

- Need to know which Objects in the Module are Requirements (shall statements)
- Need to maintain attributes for each requirement
  - Object Type – Identifies Requirements
  - Verification Method
  - Verification Class (Test Category)

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Verification Method</th>
<th>Test Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>I (Inspection)</td>
<td>First Article (FA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>Acceptance Test (AT)</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>First Article (FA)</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>Formal Qualification (QUAL)</td>
</tr>
</tbody>
</table>
RPE Region

- The Condition on the Region defines where the VCRI table will be exported in the overall flow of document generation
  - heading_number == "4.3.0-1.0-1", the Table Caption for the VCRI
- Create an additional Container which executes a new inner Object query loop to build the VCRI table based on the attributes of the Requirements
Add VCRI Region

- Add Region, Name it VCRI, add condition
Add VCRI Container

- Add VCRI Container, Add Formatted Header Rows, add Object Query for Data Rows, Add Condition for Requirements

<table>
<thead>
<tr>
<th>Container</th>
<th>1 - VCRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragraph</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Section 31</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>Requirer</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>QUAL</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Section 4</td>
<td></td>
</tr>
</tbody>
</table>

- Use Variables, Attribute Queries and Script Queries to populate Data Rows
Set Variables

- Cannot jump back to the parent heading, so store the last Heading Number and Object Heading Text as we loop through the Objects
**Set Script Expressions**

- Use Scripts to query Verification Attributes and populate Cells

```javascript
var RegExp = /^(Analysis)/; var s = Verification_Method; var i = s.search(RegExp);

if(i != -1)
    "x",
else
    "-";
```

```javascript
var RegExp = /^(Acceptance Test)/; var s = Test_Category; var i = s.search(RegExp);

if(i != -1)
    "X",
else
    "-";
```
RPE Master Pages

Master pages

A document template has no concept of pages; it is a single continuous flow. This is because a document template defines data placeholders so it is not possible to know at design time when a page will start.

But that does not mean RPE doesn’t support the page concept. RPE handles pagination through “Master Pages”. A master page is defined by a header element, a footer element and the page’s properties such as page orientation, page borders etc.

A master page is edited in the same way as you edit the template’s content with two major restrictions:

- You cannot put queries or data attributes in the master page (though the same result is possible via alternate means)
- You can add elements only inside the Header/Footer elements
Create Query and Variable for use on Masterpage

- Create new Container Element as first element in the Template, add Module Query, create Variable “moduleName” and Query the Module level Attribute “Name” to populate the Variable.
Create new Masterpage

- Create new Masterpage, set to Landscape, set the Header to display the Module Name
Set Master Page for the Container

- Set the Container to use the new Masterpage
VCRI – Word Output

- On Landscape Page
- Module Name applied to Header
- The three header rows repeat on each subsequent page

### RDM-1272

<table>
<thead>
<tr>
<th>Section 3 Paragraph</th>
<th>Title</th>
<th>Requirement ID</th>
<th>Method</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1.1</td>
<td>Heading</td>
<td>Prefix-2888</td>
<td>N/A</td>
<td>BT</td>
</tr>
<tr>
<td>3.2.1.2</td>
<td>Heading</td>
<td>Prefix-2921</td>
<td>X</td>
<td>A</td>
</tr>
<tr>
<td>3.2.1.2</td>
<td>Heading</td>
<td>Prefix-2925</td>
<td>X</td>
<td>I</td>
</tr>
<tr>
<td>3.2.1.2.1</td>
<td>Heading</td>
<td>Prefix-3548</td>
<td>X</td>
<td>D</td>
</tr>
<tr>
<td>3.2.1.2.1</td>
<td>Heading</td>
<td>Prefix-3541</td>
<td>X</td>
<td>T</td>
</tr>
<tr>
<td>3.2.1.2.1</td>
<td>Heading</td>
<td>Prefix-3542</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>3.2.1.2.1</td>
<td>Heading</td>
<td>Prefix-3543</td>
<td>X</td>
<td>AT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 4 Paragraph</th>
<th>Method</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.2.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3.2.1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3.2.1.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3.2.1.2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Not Covered in this Presentation

**Document specification**

Document templates define the workflow and content of the document while the document specification defines the document templates to be used, the concrete data sources assigned to each template data source schema and the output formats.

**NOTE** A single document specification can use any number of templates.

- Document Specifications
  - Access Multiple Data Sources
    - DOORS View as a Data Source
  - External Variables
- Master Pages
  - Headers and Footers
  - Page Numbering
  - Queries and Variable Access
- Tables and Figures – Captions, Numbering and Referencing 😞
- Table of Contents, Table of Tables and Table of Figures
- Data Source Schema
  - Dynamic Data sources
- Output Formats
  - HTML, PDF, XslFo
  - OLEs
  - Static images
- Sorting
Value of investing in RPE

- Initial investment in RPE licenses, training, and template development

- Value Gained:
  - Document generation time cut from 30 minutes to less 2-5 minutes
  - No manual steps, human error eliminated
  - Document generation schedule time reduced from several days to half a day
Daily iPod Touch giveaway

- Complete your session surveys online each day at a conference kiosk or on your Innovate 2011 Portal!

- Each day that you complete all of that day’s session surveys, your name will be entered to win the daily iPod touch!

- On Wednesday be sure to complete your full conference evaluation to receive your free conference t-shirt!