

1 EA Publishing Process

1.1 Overview

The VA Enterprise Architecture (EA) consists of static HTML sites primarily based on the data contained in Borland's CaliberRM repository. The current EA sites are listed below as well as their status as of this writing (10/15/05):

- Production EAv4.0 @ <http://10.222.67.105/eav4.0/COID-16433.html> - released on the intranet only (e.g., wwva.va.gov).
- Production EAv4.0 Summary @ <http://10.222.67.105/eav4.0%20Smry/index.html> - not released to the public.
- Development EAv4.x @ <http://10.222.67.105/forrest/COID-16433.html> - equivalent to EAv4.0.
- Staging EAv4.x @ http://10.222.67.105/forrest_stage/ - empty.

Substantial revisions to the site be communicated to the user community by a message in the welcome page. The current welcome page is generated from COID 17515 in Caliber RM.

The overall process for publishing these sites consists of two major parts, generating the textual component, as shown in *Figure 1 – Overview*, and generating the graphical component, as shown in *Figure 8 - Metis Overview*.

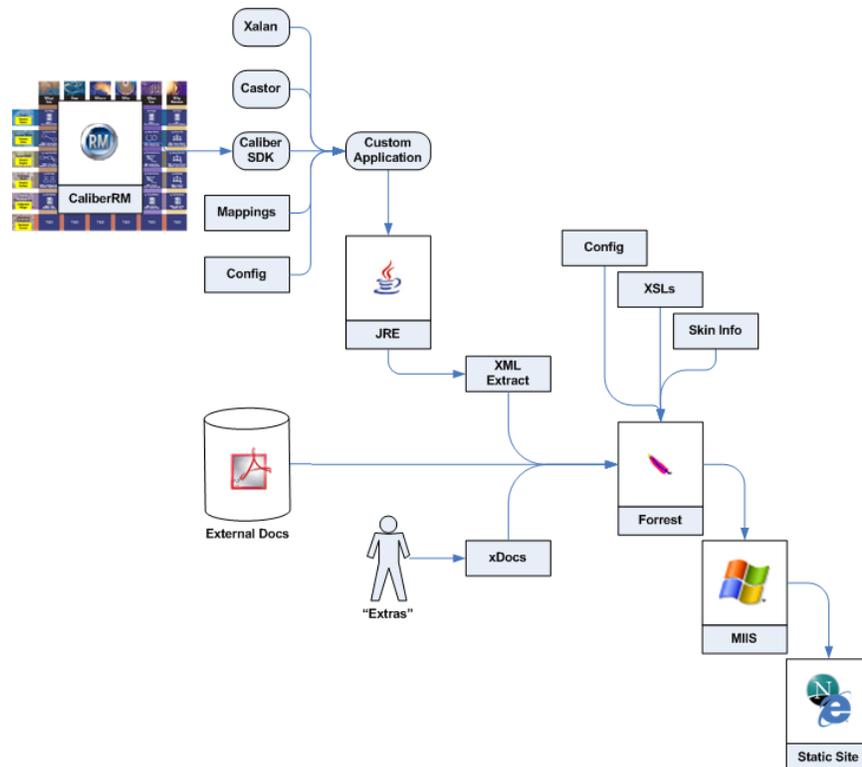


Figure 1 – Overview

The process consists of several major steps:

1. Extracting data from CaliberRM
2. Creating/refreshing several one-off documents that cannot be stored in CaliberRM
3. Generating the site using Forrest
4. Generate & exporting the Metis Model
5. Promoting the site to a new staging area for review and approval
6. Switching the staging area into the Production area.

The critical COTS applications that are independently managed and enable this process are briefly introduced below. Any key configuration or operational information related to them is highlighted throughout the rest of the document. Additionally, a reference to the Zachman Framework is listed since the organizational philosophy of CaliberRM is based on it.

1.2 Borland's CaliberRM¹

CaliberRM is a solution for managing requirements throughout the software delivery process. Designed to capture and manage business, technical, functional, and operational requirements, CaliberRM allows stakeholders across the organization to collaborate effectively so that projects are delivered on time, within budget, and to specification. Using CaliberRM to define, prioritize, and track requirements throughout the project lifecycle, software development teams can respond rapidly to ever-changing requirements without jeopardizing project success. Powerful estimation capabilities take project planning and impact analysis to a new level, helping organizations optimize the software delivery process by delivering projects with greater predictability and control.

1.3 Apache Software Foundation's Forrest²

Apache Forrest (forrest.apache.org) is a publishing framework that transforms input from various sources into a unified presentation in one or more output formats. The modular and extensible plug-in architecture is based on Apache Cocoon and relevant standards, which separates presentation from content. Forrest can generate static documents, or be used as a dynamic server, or be deployed by its automated facility.

1.4 Apache Software Foundation's Xalan³

Xalan-Java is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0 and can be used from the command line, in an applet or a servlet, or as a module in other program.

¹ Borland Software Corporation 2004. "Borland® CaliberRM™, Collaborative Requirements Management System, DELIVER PROJECTS RIGHT -- THE FIRST TIME.", Retrieved Oct 12, 2005 (http://www.borland.com/resources/en/pdf/products/caliber/crm_datasheet.pdf).

² The Forrest Community. "Apache Forrest: documentation framework", Retrieved Oct 12, 2005 (<http://forrest.apache.org/flyer.html>).

³ Apache Xalan Project. "What is Xalan?", Retrieved Oct 12, 2005 (<http://xalan.apache.org/index.html>).

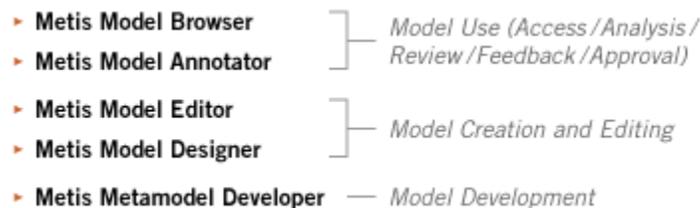
1.5 Castor⁴

Castor is an Open Source data binding framework for Java™. It's the shortest path between Java objects, XML documents and relational tables. Castor provides Java-to-XML binding, Java-to-SQL persistence, and more.

1.6 Metis Desktop⁵

Metis Desktop brings the world of advanced visualization, modeling and analysis to independent commercial enterprise and government users. This tool enables organizations to support the needs of additional users to enhance their existing EA activities.

Metis Desktop is a single tool that can be easily tailored to meet the needs of the following five roles:



Collectively, these products are known as Metis Client Tools. Each higher level of the product includes the functionality of the preceding level. For example, Designer contains of the features of Editor, Annotator and Browser.

1.7 Zachman Framework⁶

The Zachman Framework is a matrix of 36 cells covering the Who, What, Where, When, Why, and How questions of an enterprise. The enterprise is then split that into six perspectives, starting at the highest level of business abstraction going all the way down to implementation. Such objects or descriptions of architectural representations are usually referred to as Artifacts. The framework can contain global plans as well as technical details, lists and charts. Any appropriate approach, standard, role, method or technique may be placed in it.

Although frequently looked at as a framework for building computer systems, the Zachman Framework is actually a classification scheme for descriptive representations of the enterprise as a whole, irrespective of its use of computers. Through the Zachman Institute for Framework Advancement (ZIFA) (<http://www.zifa.com>), Zachman works with many users and vendors to promote the use of enterprise architecture.

⁴ The Castor Community. “The Castor Project: Quick Description”, Retrieved Oct 12, 2005 (<http://www.castor.org/index.html>).

⁵ Troux Technologies, Inc. 2005. “EA tools for the power user”, Retrieved Oct 24, 2005 (http://www.troux.com/products/eap/ea_modeling/metis_desktop.asp).

⁶ Telelogic/Popkin Software. “ZACHMAN IN BRIEF”, Retrieved Oct 13, 2005 (http://government.popkin.com/frameworks/zachman_framework.htm).

2 Data Extraction

The process begins by extracting a series of requirements from CaliberRM using a custom written Java application and marshalling them to individual XML files. The list of requirements to extract is controlled by the XML file: `ConfigiFile.xml`, with the schema shown below in *Figure 2 - CaliberRM Extract - Control File*.

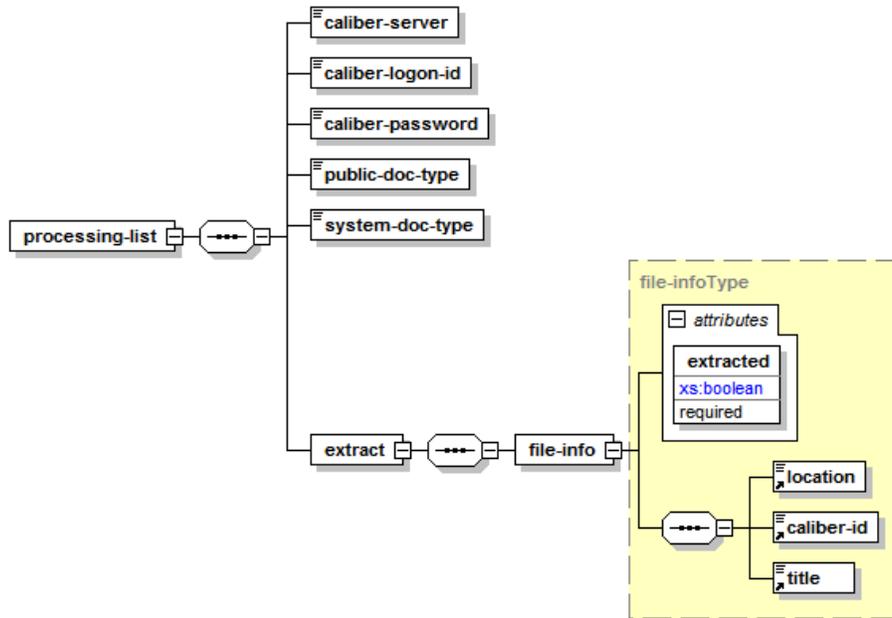


Figure 2 - CaliberRM Extract - Control File - Schema

Field	Description
caliber-id	The Caliber Id that uniquely identifies this requirement.
caliber-logon-id	The CaliberRM user Id used to logon, currently <i>JavaPgm</i> .
caliber-password	The password for the caliber-logon-id, currently for <i>JavaPgm</i> is <i>remotePWD</i>
caliber-server	IP address, Server Name, or DNS entry for the computer running the CaliberRM software, currently 10. 222. 67. 105.
file-info/@extracted	True/false indicator so that the whole site does not need to be extracted.
location	The place where this program writes it outputs this file.
public-doc-type	The public document type for the outputted XML files, currently <i>-//VACO//DTD CaliberRM V3.0//EN</i> . <div style="border: 1px solid orange; padding: 5px; text-align: center;"> <p>Warning: the value of this doc-type is tied to the site-map, which is explained in the section, <i>Transforming -//VACO//DTD CaliberRM V3.0//EN</i>. It informs Forrest how to translate these XML files into xDoc format.</p> </div>

system-doc-type	The system document type for the outputted XML files, currently: E:\Forrest\ea4\src\documentati on\resources\schema\cal i berRM- v3.dtd
title	The HTML page title that will go on the site for this information.

This file is un-marshaled using Castor to a collection of JavaBeans based on the Castor - I nput Mappi ng.xml file. The file format is explained on the Castor site (<http://www.castor.org/xml-mapping.html>) and the current file contains:

```
<?xml versi on="1.0" encodi ng="UTF-8"?>
<!DOCTYPE mappi ng SYSTEM "E:\JBProj ects\3party_util\s\castor-0.9.7\doc\mappi ng.dtd">
<mappi ng>
  <cl ass name="gov.vaea.cal i ber.cl asses.ProcessLi st">
    <map-to xml ="processi ng-li st"/>
    <fi el d name="cal i berServer" type="j ava.Lang.Stri ng"/>
    <fi el d name="cal i berLogonId" type="j ava.Lang.Stri ng"/>
    <fi el d name="cal i berPassword" type="j ava.Lang.Stri ng"/>
    <fi el d name="publ icDocType" type="j ava.Lang.Stri ng"/>
    <fi el d name="systemDocType" type="j ava.Lang.Stri ng"/>
    <fi el d name="fi les" type="gov.vaea.cal i ber.cl asses.Fi lel nfo"
  collecti on="vector">
    <bi nd-xml name="fi le-i nfo" l ocati on="extract" />
  </fi el d>
</cl ass>
  <cl ass name="gov.vaea.cal i ber.cl asses.Fi lel nfo">
    <fi el d name="ti tle" type="j ava.Lang.Stri ng"/>
    <fi el d name="cal i berId" type="i nteger"/>
    <fi el d name="l ocati on" type="j ava.Lang.Stri ng"/>
    <fi el d name="extracted" type="bool ean">
      <bi nd-xml node="attri bute"/>
    </fi el d>
  </cl ass>
</mappi ng>
```

Table 1 - Castor Input Mapping

For each individual requirement, the application uses the CaliberRM SDK to walk their requirement-node-tree and creates a collection of JavaBeans that represent the relevant data by traversing the following objects: CaliberServer, Session, Projects, Project, Baselines, Baseline, Requirements, and finally Requirement.

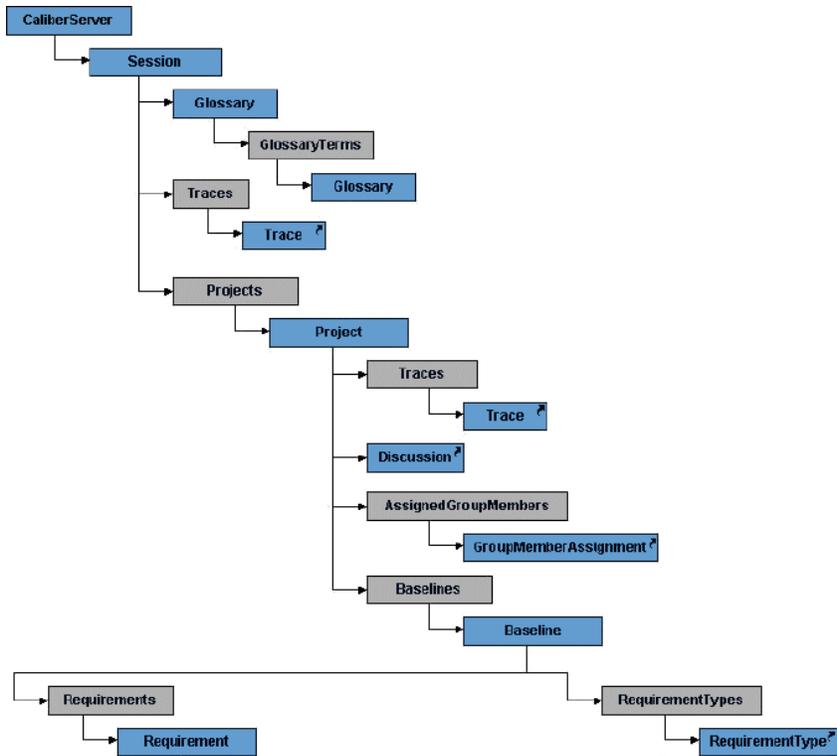


Figure 3 - CaliberRM Object Model: Root⁷

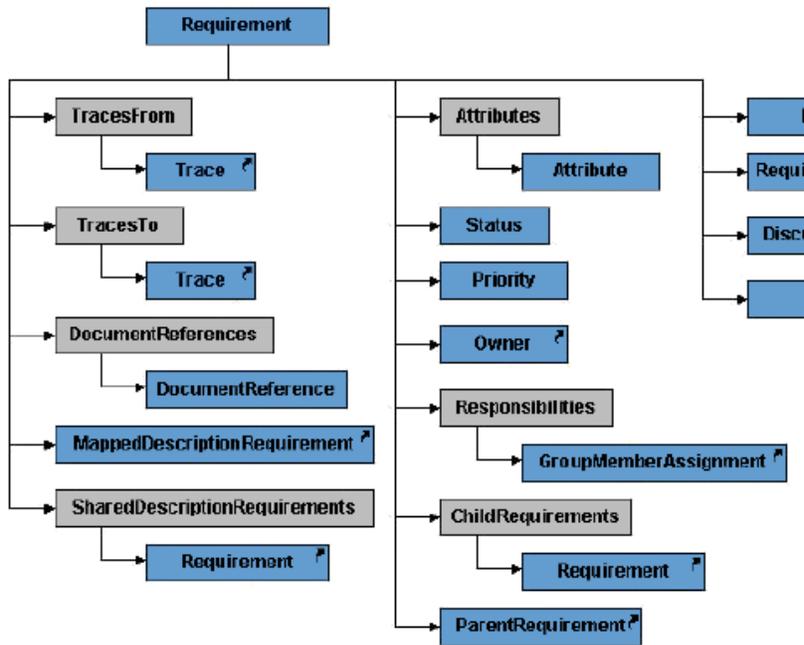


Figure 4 - CaliberRM Object Model: Requirement⁸

⁷ Borland Software Corporation 2004. "CaliberRM SDK Programmer's Guide", Appendix A: CaliberRM Object Model, page 52

⁸ Ibid., page 53

The JavaBeans are then marshaled out to their locations based on the Castor - Output Mapping.xml file. The file format is explained on the Castor site (<http://www.castor.org/xml-mapping.html>) and the current file contains:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapping SYSTEM "E:\JBProjects\3party_util\castor-0.9.7\doc\mapping.dtd">
<mapping>
  <class name="gov.vaea.caliber.classes.MyRequirement" verify-constructable="false">
    <map-to xml="requirement"/>
    <field name="internalOnly" type="boolean">
      <bind-xml node="attribute"/>
    </field>
    <field name="procurementSensitive" type="boolean">
      <bind-xml node="attribute"/>
    </field>
    <field name="isShared" type="boolean">
      <bind-xml node="attribute"/>
    </field>
    <field name="tag" type="string">
      <bind-xml node="attribute"/>
    </field>
    <field name="title" type="string">
      <bind-xml node="attribute"/>
    </field>
    <field name="coid" type="com.starbase.caliber.CaliberObjectID"/>
    <field name="name" type="string"/>
    <field name="metisObjType" type="string"/>
    <field name="docReferences" type="string" collection="arraylist" get-
method="getDocReferences"/>
    <field name="description" type="string"/>
    <field name="traces" type="gov.vaea.caliber.classes.MyRequirement"
collection="map" get-method="getTraces"/>
    <field name="requirements" type="gov.vaea.caliber.classes.MyRequirement"
collection="arraylist" get-method="getRequirements">
      <bind-xml name="requirement" location="requirements"/>
    </field>
  </class>
</class name="com.starbase.caliber.CaliberObjectID" auto-complete="true" />
</mapping>
```

Table 2 - Castor Output Mapping

The resulting output files are recursively defined requirement lists (e.g., hierarchical) conforming to the schema outlined below in *Figure 5 - CaliberRM Schema*.

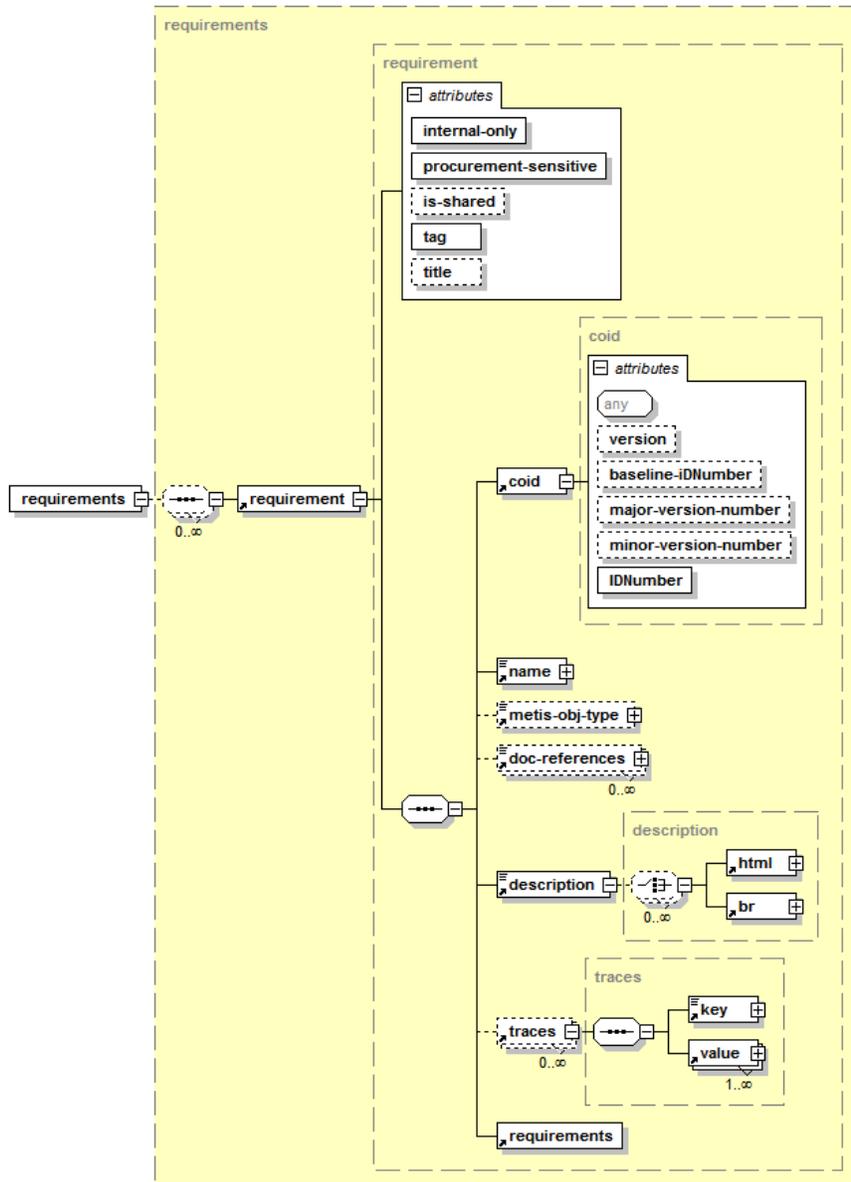


Figure 5 - CaliberRM Schema

Field	Description
coid	Caliber object id = CaliberRM's Requirement id
name	CaliberRM's Requirement description
metis-obj-type	Used to map this requirement to a Metis object type defined by AI's metamodel, reference: http://10.222.67.105/forrest/process/modeling/metamodel.html .
doc-reference	Collection of URLs that supply supporting information to this requirement, typically stored in the External Document library.
description	CaliberRM's description encoded in xHTML.
traces	CaliberRM's trace fields representing links between the requirements, we ignore the trace's direction and treat each on bidirectional.

traces/key	CaliberRM's Requirement Type field, representing the Zachman cell used to group the traces.
traces/value	Recursively defined requirement.
requirements	Recursive list of child requirements (e.g., hierarchical children).

3 One-off Documents

Although the vast majority of data is extracted from the CaliberRM repository, there are several documents outlined in this section that must be generated and modified by hand.

3.1 Production EAv4.0 Summary

Page Title	Description
Home Page	Forrest/Cocoon xDoc v2.0 encoding with the text entered by hand. Reference Link: http://10.222.67.105/eav4.0%20Smry/index.html
As-Is Tab Introduction	Forrest/Cocoon xDoc v2.0 encoding with the text entered by hand. Reference Link: http://10.222.67.105/eav4.0%20Smry/asis/index.html

3.2 Production EAv4.0

Page Title	Description
EA Acronyms	Forrest/Cocoon xDoc v2.0 Encoding of the following files: <ul style="list-style-type: none"> • acronyms.xml This file is manually produced based on exporting them from CaliberRM using the Framework Administrator: File / Admin... / Glossaries / Select <i>EA v3.0 Acronyms</i> / Modify... / Export. Then, manually manipulating it into the xDoc v2.0 table format. Reference Link: http://10.222.67.105/forrest/kb/reference/acronyms.html
EA and the FEA-PMO Initiative...BRM Mapping	HTML Encoding made up of the following files: <ul style="list-style-type: none"> • VA 2 BRM Vi ewer. i html – creates the window to the large cross reference file • VA 2 BRM Xref. htm – large cross-reference in HTML • VA 2 BRM Xref. pdf – PDF version of the cross-reference • VA 2 BRM Xref. x l s – MS Excel version of the cross-reference See the description under <i>Mandate to VA Cross Reference</i> . Reference Link: http://10.222.67.105/forrest/process/fea/VA%20%20BRM%20Vie wer.html
EA and the FEA-PMO Initiative...Pattern Registry	HTML Encoding made up of the following files: <ul style="list-style-type: none"> • Pattern Regi stry Vi ewer. i html – creates the window to the large cross reference file • Pattern Regi stry. htm – large cross-reference in HTML • Pattern Regi stry. pdf – PDF version of the cross-reference

	<ul style="list-style-type: none"> • Pattern Registry.xls – MS Excel version of the cross-reference <p>See the description under <i>Mandate to VA Cross Reference</i>.</p> <p>Reference Link: http://10.222.67.105/forrest/process/fea/Pattern%20Registry%20Viewer.html</p>
EA and the FEA-PMO Initiative...SRM Mapping	<p>HTML Encoding made up of the following files:</p> <ul style="list-style-type: none"> • VA 2 SRM Viewer.html – creates the window to the large cross reference file • VA 2 SRM Xref.htm – large cross-reference in HTML • VA 2 SRM Mapping.pdf – PDF version of the cross-reference • VA 2 SRM Xref.xls – MS Excel version of the cross-reference <p>See the description under <i>Mandate to VA Cross Reference</i>.</p> <p>Reference Link: http://10.222.67.105/forrest/process/fea/VA%202%20SRM%20Viewer.html</p>
EA and the FEA-PMO Initiative...SRM Registry	<p>HTML Encoding made up of the following files:</p> <ul style="list-style-type: none"> • SRM Registry Viewer.html – creates the window to the large cross reference file • SRM Registry Xref.htm – large cross-reference in HTML • SRM Registry Xref.pdf – PDF version of the cross-reference • SRM Registry Xref.xls – MS Excel version of the cross-reference <p>See the description under <i>Mandate to VA Cross Reference</i>.</p> <p>Reference Link: http://10.222.67.105/forrest/process/fea/SRM%20Registry%20Viewer.html</p>
EA and the FEA-PMO Initiative...TRM Mapping	<p>HTML Encoding made up of the following files:</p> <ul style="list-style-type: none"> • VA 2 TRM Viewer.html – creates the window to the large cross reference file • VA 2 TRM Xref.htm – large cross-reference in HTML • VA 2 TRM Mapping.pdf – PDF version of the cross-reference • VA 2 TRM Xref.xls – MS Excel version of the cross-reference <p>See the description under <i>Mandate to VA Cross Reference</i>.</p> <p>Reference Link: http://10.222.67.105/forrest/process/fea/VA%202%20TRM%20Viewer.html</p>
EA Terms	<p>Forrest/Cocoon xDoc v2.0 Encoding of the following files:</p> <ul style="list-style-type: none"> • terms.xml <p>This file is manually produced based on exporting them from</p>

	<p>CaliberRM using the Framework Administrator: File / Admin... / Glossaries / Select <i>EA v3.0 Terms</i> / Modify... / Export. Then, manually manipulating it into the xDoc v2.0 table format.</p> <p>Reference Link: http://10.222.67.105/forrest/kb/reference/terms.html</p>
<p>EA Tools Integration</p>	<p>HTML Encoding made up of the following files:</p> <ul style="list-style-type: none"> • tools.html – creates the window to the exported Visio HTML application. • CONOPS.htm – Exported Visio HTML application • CONOPS_files directory – Supporting Files for CONOPS.htm • CONOPS.vsd – Original Visio file located the Forrest root direction under <i>mySource/CONOPs Web</i> <p>Notes:</p> <ol style="list-style-type: none"> 1. Forrest considers the CONOPS.htm & CONOPS_files as “raw content”, but due to a bug it may not all get copied to the build site and therefore must be copied manually. <p>Reference Link: http://10.222.67.105/forrest/process/modeling/tools.html</p>
<p>Mandate to VA Cross Reference</p>	<p>HTML Encoding made up of the following files:</p> <ul style="list-style-type: none"> • Z16 to VA Viewer.html – creates the window to the large cross reference file • Z16 to VA Xref.htm – large cross-reference in HTML • Z16 to VA Xref.pdf – PDF version of the cross-reference • Z16 to VA Xref.xls – MS Excel version of the cross-reference <p>Because the cross-reference matrix is so large, they are displayed in a scrollable window (e.g. HTML’s <i>iframe</i>) with supporting PDF and Excel versions. Currently, the cross-reference table is created in Excel using extracted XML with Pivot table functionality. However, Wendel Yale is working on a SQL query that should simplify this process.</p> <p>Reference Link: http://10.222.67.105/forrest/analysis/models/Z16%20to%20VA%20Viewer.html</p>
<p>Meta-Model Cell Details</p>	<p>Forrest/Cocoon xDoc v2.0 Encoding of the following files:</p> <ul style="list-style-type: none"> • metamodel.xml – contents of the page generated by <i>Domain Extract - Forrest.xslt</i> (see Supporting files below). • images/al's_small_metamodel.gif – the small opening 6x6 graphic exported from Al’s Visio Metamodel • images/al's_big_metamodel.jpg – the large 6x6 graphic exported from Al’s Visio Metamodel • images/Zxy-Domain.gif (36 in total) – where <i>x</i> is the Zachman Row ranging from 1-6 and <i>y</i> is the Zachman Column, also ranging from 1-6.

	<p>Supporting files:</p> <ul style="list-style-type: none"> • Al Zuech’s Visio Metamodel – contains Al’s notional or cartoon view of the Metamodel used to organize the repository; named similar to ZachmanModelingV04-ao20050218.vsd. • Domain Extract - Forrest.xslt – creates the metamodel.xml based on the information within the actual Metis metamodel, located in the Forrest root direction under <i>mySource\Mappings\Mapping Metis to Visio Metamodel</i> • Metis’s metamodel - ea-domain.kmd - located in the default metis installation directory: Metis3.4\xml\http\xml.vaea.gov\xml\metamodels\ <p>Notes:</p> <ol style="list-style-type: none"> 1. The Visio metamodel 6x6 graphics must be cleaned up by deleting the inter cell links and some of the reminder notes. 2. Each of the individual cell images (e.g. Zxy-Domain.gif) must be cut out of the larger al’s big metamodel.jpg and saved in the image directory by Domain. This task is truly painful, the best approach that I’ve found is printing the Visio to large paper sized PDF file, then cutting the individual cell out of the PDF, cleaning them up in an image editor, and finally saving them individually. 3. Embedded in Forrest.xslt is an image map that hard-codes bookmarks of the cell’s description to the area of al’s small metamodel.gif that contains the corresponding slice. <p>Reference Link: http://10.222.67.105/forrest/process/modeling/metamodel.html</p>
Metis Web	Generating the various Metis HTML reports that represent the “graphical” view of the Zachman Framework is outside the scope of this document.

3.3 Image Maps

This section is drawing attention to another hack that enables images imbedded in CaliberRM to be clickable (e.g., HTML’s hotspots). The file *image_maps.xml* located in the Forrest root under *src\documentati on\resources\styl esheets* works in conjunction with the *caliber2documentv13-v3.xsl* that transforms the CaliberRM XML to Forrest’s xDoc v2.0 XML and will be described in the next section. Regardless, of its processing mechanics, the contents of the file are manually generated. The structure of the file, shown in *Figure 6 - Image Map Schema*, mirrors HTML’s [Image Map](#) tag where the Map’s name attribute signifies the CaliberRM’s Requirement Id that contains the image that the map applies.

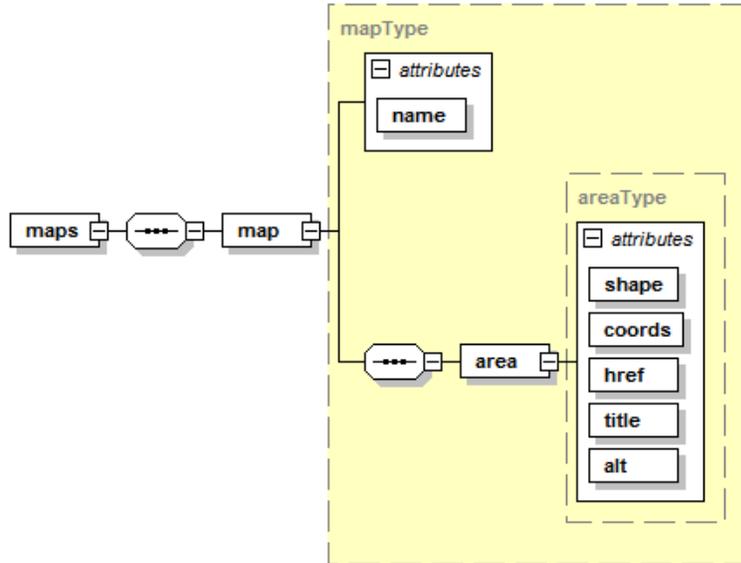


Figure 6 - Image Map Schema

For instance *Table 3 - Image Map Example* shows a code snippet where Requirement (EADOCS) 16862 contains a picture, shown in *Figure 7 - CaliberRM Screen Shot*, with two rectangular hotspots (1) Associated and Allied Health Education; (2) Graduate Medical Education that will be hyperlinked (via hotspots and bookmarks) to CaliberRM’s Requirement 17180 and 17181, respectively.

```

<map name="16862">
  <area shape="rect" coords="205, 341, 378, 405" href="#oid-17180" title="Goto
Associated and Allied Health Education" alt="Goto Associated and Allied Health
Education"/>
  <area shape="rect" coords="213, 174, 362, 236" href="#oid-17181" title="Goto Graduate
Medi cal Education" alt="Goto Graduate Medi cal Educati on"/>
</map>

```

Table 3 - Image Map Example

Warning: Since only one name can be associated with a map, then only one image can be included in the specific requirement. If multiple images per requirement are needed, artificially create child requirements and place the images there.

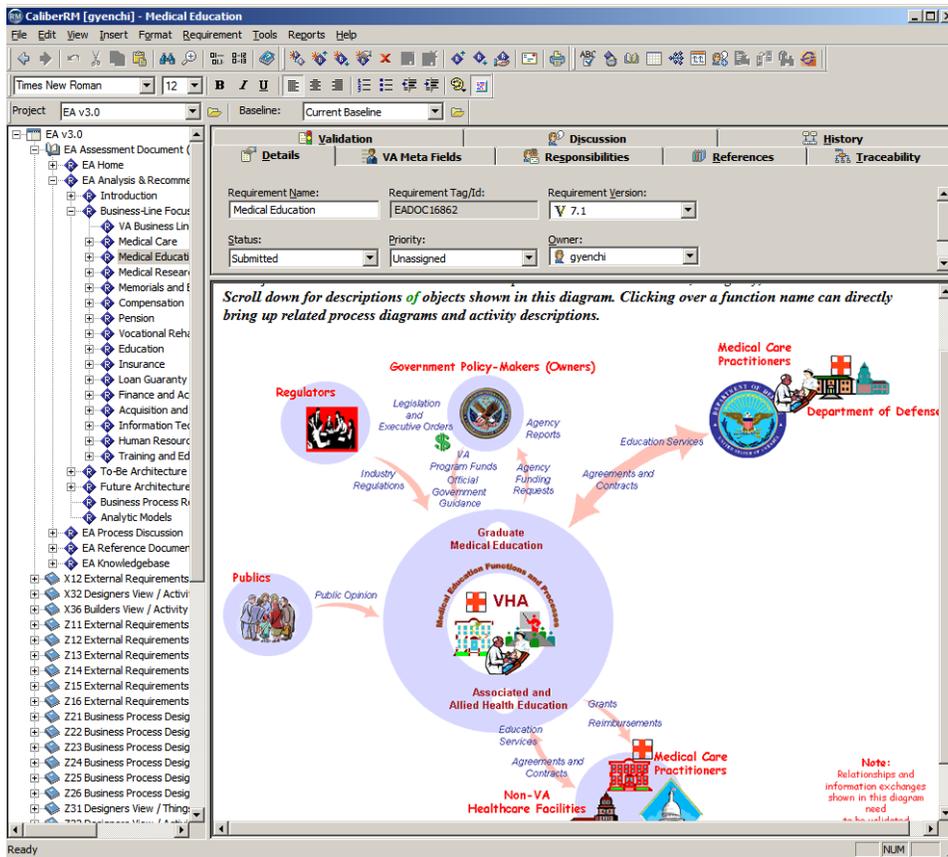


Figure 7 - CaliberRM Screen Shot

4 Forrest

Forrest is the open-sourced product that's used to generate the EA sites. Generically, the following Forrest pages document the background needed to understand the product:

- Welcome - <http://forrest.apache.org/index.html>
- Downloading - <http://forrest.apache.org/mirrors.cgi>
- Getting Started - http://forrest.apache.org/docs_0_60/your-project.html
- Support - <http://forrest.apache.org/mail-lists.html>

Note:

Al Zuech has a technical requirement that states all EA materials must run from a standalone CD under a standard browser configuration with no client-side software installations or individually licensed supporting software. Therefore, we use Forrest to generate the [static site](#) and, to the best of our ability, copy all externally referenced document to a local directory under the control of Forrest. We cannot use any server-side processing (e.g., scripting, databases, etc.) since a standalone user would not have access to those resources.

Table 4 - CD Requirement

Specifically to the EA site, the following sections highlight specific topics where we customized Forrest to meet the VA's needs.

4.1 Transforming -//VACO//DTD CaliberRM V3.0//EN

The data extracted from CaliberRM is stored in a XML files described by the document type: -//VACO//DTD CaliberRM V3.0//EN. Using XSLTs, the CaliberRM document type is transformed into Forrest's -//APACHE//DTD Documentation V2.0//EN. Generically, the following Forrest pages document this topic:

- http://forrest.apache.org/docs_0_70/your-project.html#adding_new_content_type
- http://forrest.apache.org/docs_0_70/cap.html
- http://forrest.apache.org/docs_0_80/validation.html#Validating+new+XML+types
- <http://xml.apache.org/commons/components/resolver/resolver-article.html>
- http://forrest.apache.org/docs_0_80/howto/howto-custom-html-source.html

Specifically to the EA site, the sitemap.xmap shown below directs the processing of the -//VACO//DTD CaliberRM V3.0//EN document type to the caliber2documentv13-v3.xsl. Note the *Path* variable being sent into the xsl file, this variable is very important because it's used to generate the *\$root* variable using the `../../../../skins/common/xslt/html/dotdots.xsl`. The *\$root* variable contains the relative reference back to the root directory that's used to resolve many of the image and external document addresses due to *Table 4 - CD Requirement*.

```

<map: components>
  <map: actions>
    <map: action logger="sitemap.action.sourcetype" name="sourcetype"
src="org.apache.forrest.sourcetype.SourceTypeAction">
      <sourcetype name="caliber-v3.0">
        <document-declaration public-id="-//VACO//DTD CaliberRM
V3.0//EN"/>
          </sourcetype>
        </map: action>
      </map: actions>
    <map: selectors default="parameter">
      <map: selector logger="sitemap.selector.parameter" name="parameter"
src="org.apache.cocoon.selector.ParameterSelector"/>
    </map: selectors>
  </map: components>
  <map: resources>
    <map: resource name="transform-to-document">
      <map: act type="sourcetype" src="{src}">
        <map: select type="parameter">
          <map: parameter name="parameter-selector-test"
value="{sourcetype}"/>
            <map: when test="caliber-v3.0">
              <map: generate
src="{project:content.xdocs}{../../../../1}.xml"/>
              <map: transform
src="{project:resources.stylesheets}/caliber2documentv13-v3.xsl">
                <map: parameter name="path"
value="{../../../../1}"/>
              </map: transform>
            </map: when>
          </map: select>
        </map: act>
      </map: resource>
    </map: resources>
  
```

Table 5 - Site Map Snippet

Outside of the `sitemap.xmap`, the other important files that enable the transformation include:

- **`project.schema-dir/catalog.xcat`** – used to add the `caliberRM-v3.dtd` to the Forrest catalog.
- **`project.schema-dir/caliberRM-v3.dtd`** - used to describe the CaliberRM XML structure.
- **`project.stylesheets-dir/caliber2documentv13-v3.xsl`** – used to translate the requirements portion to xDoc v2.0 format.
- **`project.stylesheets-dir/caliberdescription2documentv13.xsl`** – used to translate the requirement's xHTML description to xDoc v2.0 format.
- **`project.stylesheets-dir/image_maps.xml`** – used to map areas of a requirement's image to hyperlinks via HTML hotspots (see also *Image Maps*).
- **`project.skins-dir/common/xslt/html/dotdots.xsl`** – used to create the `$root` variable.

4.2 Location Map

The location map is used to copy external documents to local directories (see *Table 4 - CD Requirement*). Generically, the following Forrest pages document this topic:

- http://forrest.apache.org/docs_0_80/locationmap.html
- <http://www.mail-archive.com/user@forrest.apache.org/msg01212.html>

Specifically to the EA site, the source files for these external documents are stored on the EA production server (`vacodeve5`) under the shared folder named *Public*, as follows:

- EA v4.0 @ `\\10.222.67.105\Public\webroot\ea3`
- EA v4.x @ `\\10.222.67.105\Public\webroot\ea4x`

These folders are addressable from two Virtual Directories:

- <http://10.222.67.105/open/ea3/>
- <http://10.222.67.105/open/ea4x/>

Hence, the `locationmap.xml` and `sitemap.xmap` snippets to produce EA v4.0 are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<locationmap xmlns="http://apache.org/forrest/locationmap/1.0">
  <components>
    <matchers default="1m">
      <matcher name="1m"
src="org.apache.forrest.locationmap.WildcardLocatorMapHi ntMatcher"/>
    </matchers>
  </components>
<!--
sitemap.xmap and http://www.mail-archi ve.com/user@forrest. apache.org/msg01199.html
-->
<locator>
  <match pattern="extDocs/**.pdf">
    <location src="http://10.222.67.105/open/ea3/{1}.pdf"/>
  </match>
</locator>
</locationmap>
```

<pre> </match> </locator> </locationmap> </pre>
<pre> <map:match pattern="extDocs/**/*.pdf"> <map:read src="{1m:extDocs/{1}.pdf}" mime-type="application/pdf" /> </map:match> </pre>

Table 6 - Location & Site Map Snippets

However, a close examination of the CaliberRM data will show no references to the folder *extDocs*. This is a hack to trick Forrest into making the external documents locally controlled by translating the 10.222.67.105 link into *extDocs* during the *-//VACO//DTD CaliberRM V3.0//EN* into *-//APACHE//DTD Documentation V2.0//EN* doc-type translation. Specifically this transformation is located in *caliber2documentv13-v3.xsl* under the *requirement* template for the *doc-references* tag and in *caliberdescripti on2documentv13.xsl* under the *a* (e.g. Anchor) template for the *@href* attribute.

<pre> <xsl:when test="starts-with(., 'http://10.222.4.39/open/eav3/')"> <xsl:value-of select="\$root" /> <xsl:text>extDocs</xsl:text> <xsl:value-of select="substring-after(., 'http://10.222.4.39/open/eav3')" /> </xsl:when> <xsl:when test="starts-with(., 'http://10.222.67.105/open/eav3/')"> <xsl:value-of select="\$root" /> <xsl:text>extDocs</xsl:text> <xsl:value-of select="substring-after(., 'http://10.222.67.105/open/eav3')" /> </xsl:when> </pre>
--

Table 7 - extDocs Snippet

4.3 VACO-EA Skin

Forrest uses skins to change the graphical appearance of the site; attempting to make it compliant with the following VA guidance:

- [Directive 6102](#) - establishes policy for the Department of Veterans Affairs (VA) employees in managing, maintaining, establishing, and presenting information on VA’s Internet / Intranet Service Sites and use of related Internet services (hereafter referred to as “Internet”).
- [Handbook 6102](#) - establishes Department-wide procedures for managing, maintaining, establishing, and presenting VA Internet/Intranet Service Sites or related services (hereafter referred to as “Internet”). This handbook implements the policies contained in VA Directive 6102, Internet/Intranet Services. This includes but is not limited to File Transfer Protocol (FTP), Hypertext Markup Language (HTML), Simple Mail Transfer Protocol (SMTP), Web pages, Active Server Pages (ASP), e-mail forums, and list servers.
- [Handbook 6102, change 1](#) – modifies Handbook 6102 to clarify VA’s cookie use, Section 508 guidelines, add guidance on posting of Hot Topics, update approved warning notices, and include minor editorial changes.

Generically, the following Forrest pages document this topic:

- http://forrest.apache.org/docs_0_80/your-project.html#skins
- http://forrest.apache.org/docs_0_80/skins.html
- http://forrest.apache.org/docs_0_80/linking.html

Specifically to the EA site, the base skin is *Pelt*. It's customized with EA specific CSS elements in `skinconfi.g.xml` that are applied via the `cal i ber2documentv13-v3.xsl` and `cal i berdescri pti on2documentv13.xsl` transformations:

```

<extra-css>
  #top {
    background: url (i mages/BannerI mage. gi f) no-repeat;
    posi ti on: rel ati ve;
    wi dth: 100%;
  }
  di v. groupI ogo {
    margi n-top: 30px;
    margi n-left: 74px;
  }
  table.ea-tracetbl {
    border-col l apse: col l apse;
  }
  table.ea-tracetbl th, table.ea-tracetbl td {
    border: thi n sol id Gray;
  }
  th.ea-col 1 {
    wi dth: 25%;
  }
  th.ea-col 2 {
    wi dth: 75%;
  }
  ul.ea-xref {
    margi n-bottom: 15px;
  }
  p.center {
    text-al ign: center;
  }
  p.left {
    text-al ign: left;
  }
  p.right {
    text-al ign: ri ght;
  }
  h2, h3, h4, h5, h6 {
    col or: Bl ue;
    font-wei ght: bol d;
  }
  di v.secti on{
    border-left-col or: Si l ver;
    border-left-styl e: sol id;
    border-left-wi dth: 1px;
    posi ti on: rel ati ve;
    paddi ng-left: 10px;
    margi n-bottom: 15px;
  }
</extra-css>

```

Table 8 - Skin Configuration: Extra CSS

Also, the standard *Pelt* skin's XSLTs are applied with the exception of overriding the Section tag in `document2html.xsl`:

```

<xsl:template match="secti on">
  <xsl:apply-templates select="@i d"/>
  <xsl:vari abl e name="I evel " select="count(ancestor-or-sel f:: secti on)"/>

  <xsl:choose>
    <xsl:when test="$I evel =1">
      <di v cl ass="ski nconf-headi ng-1">
        <h1>
          <xsl:val ue-of select="ti tle"/>
        </h1>
      </di v>
    </when>
  </choose>

```

```

        </xsl:when>
        <xsl:otherwise>
            <div class="skinconf-heading-2">
                <h2>
                    <xsl:value-of select="title"/>
                </h2>
            </div>
        </xsl:otherwise>
    </xsl:choose>

    <div class="section">
        <xsl:apply-templates select="*[not(self::title)]"/>
    </div>
</xsl:template>

```

Table 9 - VACO-EA Skin *document2html.xml* Snippet

Warning: the *vaco-ea* skin is based on the *Pelt* skin. The *Pelt* skin along with the *Common* elements skin are defined in the directory `apache-forrest-head\forrest\main\webapp\skins`, but copied to the local project directory `project.skins-dir`. Therefore, fixes to the standard skins will not be reflected in the local project version. As the Forrest application is refreshed, the *Pelt* & *Common* element skin directories should also be compared to the local project versions and changes incorporated.

4.4 Different Sites

The same Forrest source is used to generate the three sites: Production, Summary, and To-be only. The content of each site is controlled by the three files:

- `index.xml` – Default home page
- `site.xml` – left-hand menu structure, which also controls the inclusion of content
- `tabs.xml` – the top-level tab navigation

These files are stored in the xdoc root directory as sets under:

- `_Site Mapping Files - Full`
- `_Site Mapping Files - Summary`
- `_Site Mapping Files - Summary - ToBe Only`

That each corresponds to the sites mentioned above. They must be manually copied from these directories up one level to the xdoc root to be used.

5 Metis

The graphical portion of the EA site is created using Metis's desktop tool. It produces an iconic model of the data contained in the CaliberRM Zachman cells (a.k.a. the z-cells).

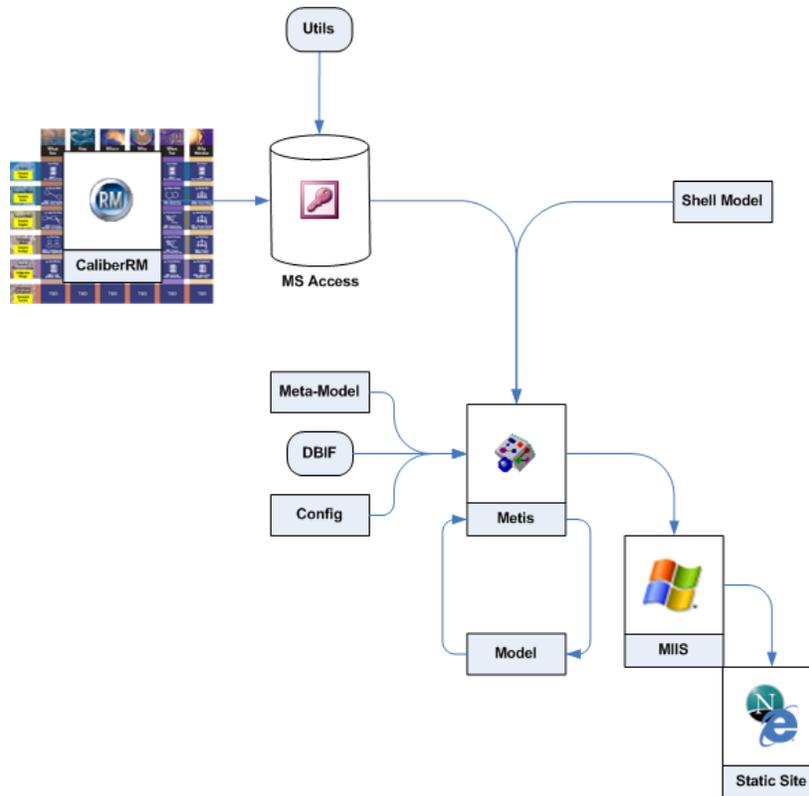


Figure 8 - Metis Overview

The process consists of several major steps:

1. Exporting CaliberRM data to MS Access
2. Running a MS Access utilities to create data relationships and views
3. Importing the Data into Metis
4. Exporting the Metis Model to HTML reports
5. Publishing the reports

5.1 Exporting CaliberRM

CaliberRM client provides functionality under *File / Export to Access...* to create an MS Access database from a project's requirements. To speed up the export process, two large tables can optionally be excluded: *RequirementChange* and *RequirementRevision*. These tables contain the requirements' revision history, which is not used in the site generation.

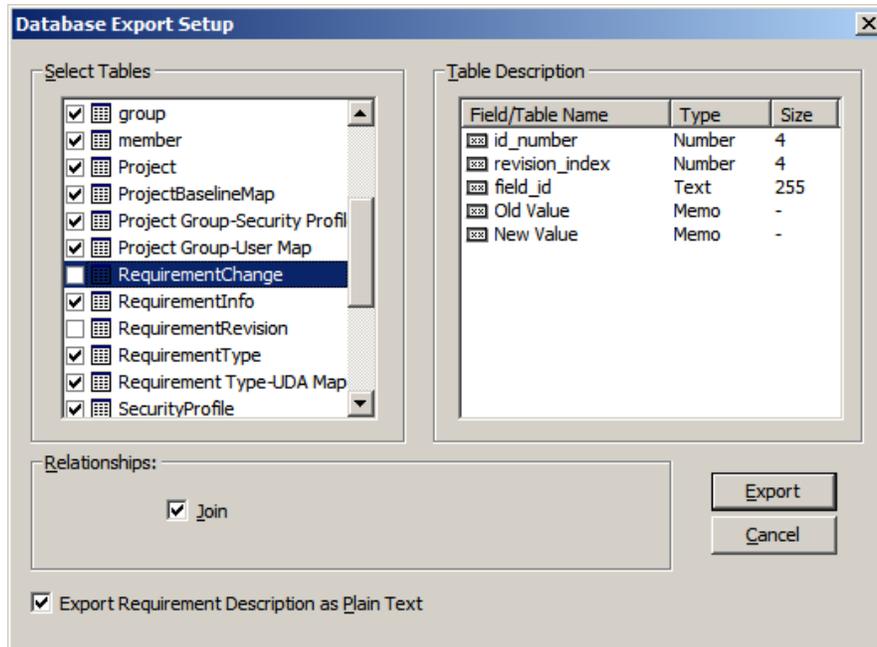


Figure 9 - CaliberRM Export Parameters

Warning: Metis’s *Graphic* tab correctly displays HTML, however their *Instance* tab does not and this is where the *description* field is displayed. Therefore, the HTML tags need to be stripped by setting the “*Export Requirement Description as Plain Text*” option, as shown above.

This limitation is a bug and they’ve committed to fixing it in an upcoming, but unspecified release, as of v3.4.6 it was still outstanding.

5.2 Creating relationships and views

To simplify the Metis import, several relationships and views need to be created using a custom written VBscript, `EA_Cal i berRM_Access_Ut i l s. bas`. This script must be imported and run in MS Access by using *Tools / Macros / Visual Basic Editor; File / Import File*; choose `EA_Cal i berRM_Access_Ut i l s. bas`; and finally run the macro `I n i t i a l i z e D B`. This macro will do the following:

- Creates and populates the *eaChild2Parent* table, which establishes a relationship between hierarchically related requirements along with listing their requirement types.
- Augments the *DocumentReferences* table with a new field *relURL* for Relative URL that replaces the “`http://10.222.67.105/open/`” with “`../..`”. See *Table 4 - CD* for a justification.
- Creates a set of queries that makes the Metis DBIF mapping file easier. Also, note that each query is duplicated with a “Debug-” prefix. The debugging queries

will execute in MS Access and MS Query, whereas the standard queries will only execute in Metis due to the differences in the wildcard characters “%” versus “*”.

5.3 Metis Import

Metis’s Database Interface (DBIF) plug-in is used to populate the data into a Metis model. Due to several Metis bugs⁹ related to container creation during the import, we’ve created a shell model `ea_shell.kmd/kmv`. This shell contains 36 containers where the data will be populated that correspond to AI’s Zachman Model & the CaliberRM requirement types.

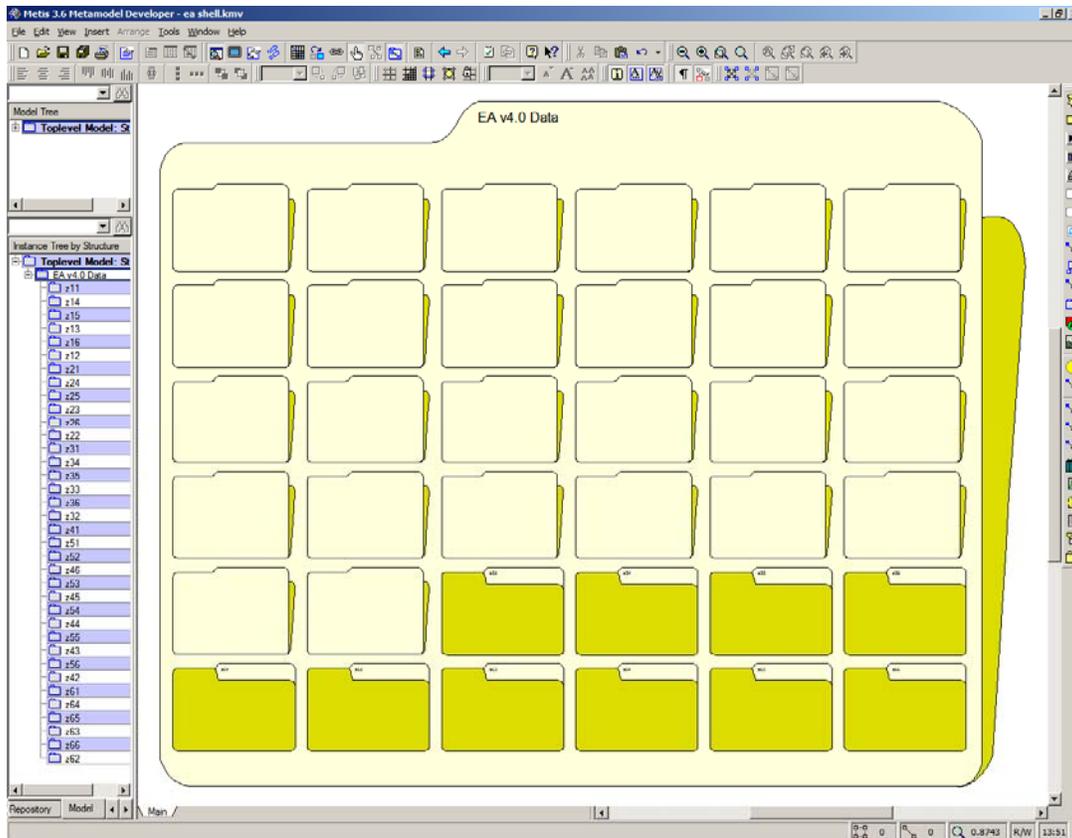


Figure 10 - Metis EA Shell

Using the File /
4 passes of the data

⁹ Per Metis Support: “[Kjell Tangen] We are aware of these issues and currently investigating how to solve them in 3.6”; however, the fixes have not been verified.

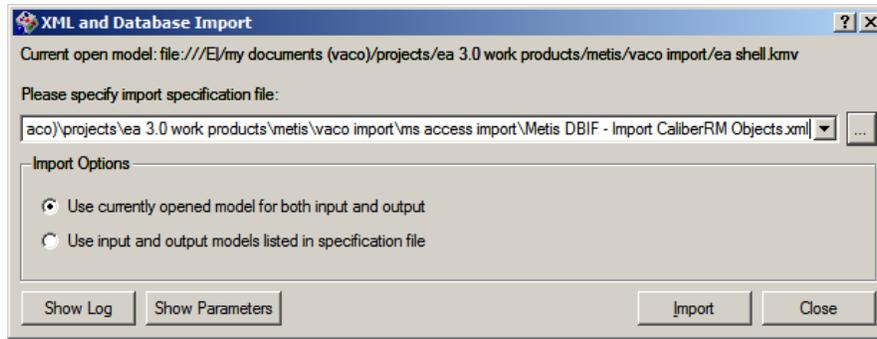


Figure 11 - Metis DB Import screen

- qEAReqInfo -
- qEATraces -
- qEAExternalRefs -
- qAssignedObjTypes -
- qPossibleObjTypes -
- qEAParentChild -

6 Publishing the Site

Physically these sites are hosted on vacodeve5 @ 10.222.67.105. Each EA “release” is stored into a new directory under the *Public* share, as shown on the left:

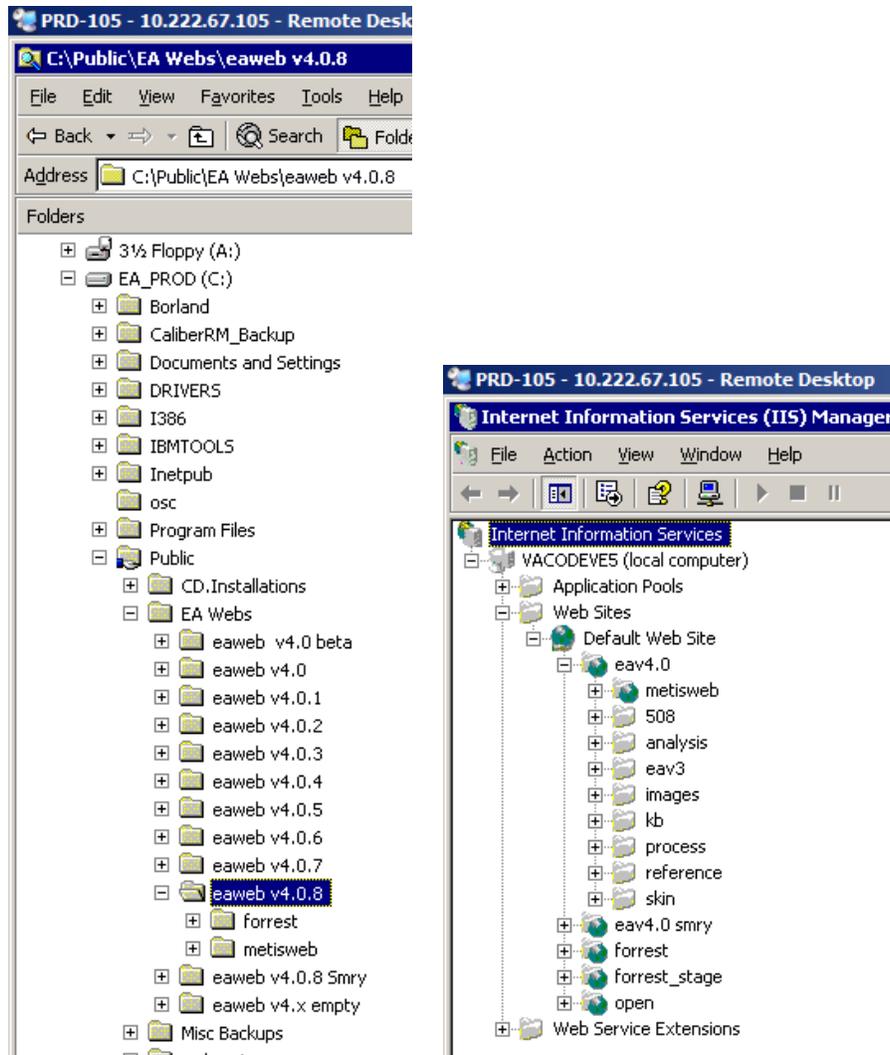


Figure 12 – File Virtual Directory Structures

Using Microsoft’s Internet Information Services (IIS) Manager, Virtual Directories have been created for each “site”, as shown on the right. Currently, Virtual Directories *eav4.0* & *eav4.0 smry* point to the physical directories *eaweb v4.0.8* and *eaweb v4.0.8 Smry*, respectively. These represent the official release of EA version 4.0. For the upcoming EA v4.x release, the Virtual Directories *Forrest* and *Forrest_stage* will be used.