

# Re-use, Re-purpose, Re-package

A General Engine Products, Inc. Case Study

John F. Terris <john.terris@vftis.spx.com>

## Abstract

General Engine Products, Inc. (**GEP**), a wholly owned subsidiary of AM General Corporation, produces Original Equipment Manufacture (**OEM**) 6.5L diesel engines, both naturally aspirated and turbo-charged, for use in military, automotive, marine, and various other industrial and commercial applications. With GM's technical support, **GEP** developed a highly efficient engine assembly operation in Franklin, Ohio, to assemble and market an improved 6.5L diesel engine to continue to satisfy the market demand. There are three main documents used by the Franklin plant to assemble engines: Process Control Plan (**PCP**), Operator Instructions, and Product Assembly Document (**PAD**).

These three manufacturing documents share elements, such as torque specifications, part numbers, tool numbers, textual instructions, and assembly order. Keeping these documents up-to-date, consistent, and accurate is very difficult because there is very little time to update all three documents and they are maintained by two groups, in three formats, in two locations.

**GEP** vision is to have a system where both manufacturing and product engineers could develop and maintain the information they are responsible for while reducing the duplicate effort, increasing the overall quality of the information, and minimizing publishing latency. Once all the information has been completed, reviewed, and approved, the system would be able to deliver all three manufacturing documents, each with a very different presentation and formatting style, on paper and over the Web directly to the assembly line operator at the plant.

This paper describes the steps **GEP** has taken to structure and develop their data using SGML/XML to put themselves in a position where they are very close to realizing that vision. It will also explore different techniques that can be used when designing systems that require:

- |            |   |
|------------|---|
| Re-Use     | using the EXACT same information object (image or text) in more than one “document”   |
| Re-Purpose | extracting and/or formatting the same piece of information in many different ways, usually producing a different document type targeted for a different user and/or purpose |

Re-Package

delivering multiple document types via different media and even to different devices

## 1. The Study

This paper assumes some level of knowledge in Arbortext's Adept/Epic products and Chrystal Software's Astoria.

### 1.1. General Engine Products, Inc.

General Engine Products Inc. (**GEP**), a wholly owned subsidiary of AM General Corporation, produces **OEM** 6.5L diesel engines, both naturally aspirated and turbo-charged, for use in military, automotive, marine, and various other industrial and commercial applications.

GEP was born out of the need to sustain the production of the 6.5L diesel engine for military and commercial use. General Motors (**GM**), opting for a higher horsepower Isuzu engine for some of its vehicle lines, chose to support the transfer of their 6.5L production knowledge and technology to General Engine Products, Inc. With **GM** technical support, **GEP** developed a highly efficient engine assembly operation in Franklin, Ohio, to assemble and market an improved 6.5L diesel engine to continue to satisfy the market demand. The 6.5 diesel will continue to be the heart of the Military HUMVEE® and the commercial HUMMER® vehicle, as well as a part of other current **OEM** products.

Production at the 92,000 sq. ft. Franklin, Ohio, assembly facility began in July 2000, reaching full production capability in March 2001.

HUMVEE® is a registered trademark of AM General Corporation

HUMMER® is a registered trademark of General Motors Corporation

### 1.2. The Documents

As **GEP** began implementing the assembly operation in Franklin, Ohio, the engineering group reviewed the documentation requirements and noticed some potential concerns. Three main types of documentation are used at the assembly plants: Product Assembly Document, Process Control Plan, and Operator Instructions. Each is described in the following sections.

#### 1.2.1. Product Assembly Document (PAD)

This document, the first to be developed, is created and maintained by the product design engineers. It contains information such as torque specifications, tool numbers, part numbers, assembly notes, and illustrations/visual aids that support each step in the assembly process. This is the primary communication method between the product design engineers and the manufacturing engineers. This document is available to the assembly line operators. However, it is difficult to control, update, and deliver to the plant floor.

### 1.2.2. Process Control Plan (PCP)

This document is developed and maintained by manufacturing engineers and contains information such as torque specifications, tool numbers, part numbers, quality checks, the assembly order, and brief textual instructions. This document serves as the International Standards Organization (ISO) control document, which means any changes to the manufacturing process MUST be reflected in this document or GEP could lose their ISO certification.

### 1.2.3. Operator Instructions

This document, usually developed after the PCP is complete, is also created and maintained by the manufacturing engineers. It contains detailed technical instructions for each step in the assembly process, including torque specifications, tool numbers, and part numbers. This document is used for the initial training of the assembly line operators. In reality, it is rarely used and usually the last document to be updated, if it is updated at all.

## 1.3. Opportunities

It should be apparent that much of the information contained in these three manufacturing documents is the same, specifically the torque specifications, part numbers, tool numbers, textual instructions, and assembly order. Keeping these three documents up-to-date, consistent, and accurate is very difficult because

- a. timing is critical in a manufacturing environment, which means there is very little time to update all three documents, and
- b. the documents are maintained by two different groups, in three different formats, and in two different locations.

Keep in mind that it is very common during the manufacturing cycle of a product for part numbers or tool numbers to change or new special tools to be developed to enhance a process. Even the assembly process itself may change if the manufacturing engineers feel the changes would increase quality and/or make the process more efficient. In addition to those opportunities, GEP felt that the format of the PAD was a problem for two reasons:

- It was not easy to find the desired information because the PAD were organized by Universal Product Codes (UPC), which typically means nothing to the line operators.
- Many of the assembly procedures for different engines were identical, with the exception of part numbers and, in some cases, tools and torque specifications. Because the PAD's put all the variants in one document, the line operator had the extra burden of identifying the appropriate information for the engine they were assembling.

Other factors that caused GEP to revisit the assembly documentation process were:

- Cycle Time Increase – The assembly plant is divided into three lines: base engine, fuel, and trim. Each line is divided into stations. One or more line operators work at each station

and have one or more sequences or steps to perform. “Cycle Time” is defined as the duration that the engine is at any given station. In other words, it is how long the line operators have to do their job. Typically, cycle times in assembly plants are one or two minutes, maybe even less. In that case, each line operator would have only a few tasks to complete, such as “select correct seat and put it in the back of the car”. The Franklin plant was going to run at about a 10-minute cycle time. Each station would have many tasks to remember and to complete within those 10 minutes. Getting the appropriate assembly instructions quickly would be imperative.

- Low Volume, High Number of Products/Variants – GEP production requirements fall into a low volume, high variance category. Each year, GEP was planning to offer 20 or more different engine type/variant combinations. To the line operator, this meant that he or she might build a particular engine only once a week and work on four to five different engines in a day. It would be very rare for the exact same engine to come down the line twice in a row.

With all of these opportunities and factors, it was obvious to GEP that a new process of creating, publishing, and distributing manufacturing documentation was needed.

## 1.4. The GEP Vision

GEP vision is to have a system where both manufacturing and product engineers could develop and maintain the information they are responsible for while reducing the duplicate effort, increasing the overall quality of the information, and minimizing publishing latency. Once all the information has been completed, reviewed, and approved, the system would be able to deliver all three manufacturing documents, each with a very different presentation and formatting style, on paper and over the Web directly to the assembly line operator at the plant.

The paper delivery would be customized for each engine and organized to match the manufacturing process (engine type, line, station, and steps). The electronic delivery would work in conjunction with the barcode on the engine manifest. As the engine entered the station, the barcode would be scanned. Based on the engine type and the line/station where it was scanned, the correct information would be assembled and delivered to the line operator. The delivery platform would consist of a wireless network in the plant where each line operator would have his or her own handheld or hands-free device to display the information.

The system would also feed the future GEP Service Information Publishing System as well as support a “feedback” or “correction” system for documentation errors found by the end user.

This vision is certainly not revolutionary in the publishing and/or SGML/XML industry. However, in this author’s opinion, it is not very widespread in the manufacturing industry. The vision is certainly very revolutionary to GEP and automotive manufacturing.

## 2. GPAS

### 2.1. Analysis

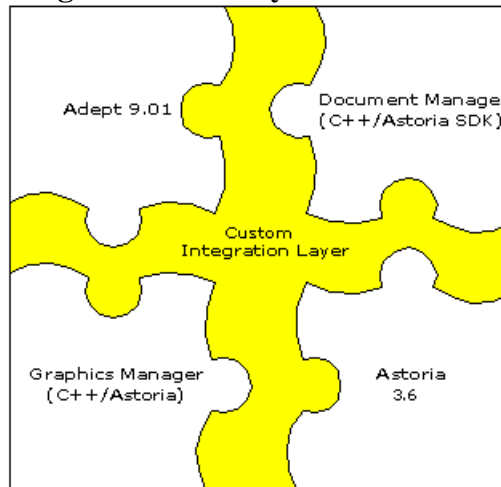
Once [GEP](#) shared their vision with Valley Forge, we entered into a very detailed “information analysis”. Many people in the industry call this document analysis, but I like to call it information analysis because we do not limit the analysis to just documents or publication types. We look at processes, uses of the data, relationships in the data, sources from which the data is developed, and so on. It was obvious during the definition phase of the project that 80 percent of the information elements necessary to produce the current three publication types ([PCP](#), [PAD](#), Operator Instructions) existed in the [PAD](#). For this reason, and the fact that the [GEP](#) champion of the project was responsible for [PAD](#), we decided to focus on the [PAD](#) and let the other publication types fall out later. During the information analysis phase, we identified all the information elements and processes required to meet the following objectives:

- Allow the line operator to successfully perform his or her job.
- Allow all end users to locate the information.
- Allow all [GEP](#) administrators to effectively manage variants and revisions.
- Allow continuous, easy, and seamless updates during the life of a [PAD](#).
- Allow information created once to be used many times.
- Allow all [GEP PAD](#) writers to locate graphics during development.
- Allow all [GEP PAD](#) writers to copy an existing [PAD](#) to assist in making other [PAD](#).

### 2.2. System Overview

The result is a system called [GPAS](#) (*General Engine Products PAD Authoring System*). [GPAS](#) was developed using customized off-the-shelf software, as shown in [Figure 1](#)

**Figure 1. GPAS System Overview**



In addition to the base authoring and development system, an entire composition (publishing) module and Web-based viewer were created.

### **2.3. Authoring Environment and Infobase (Repository)**

Once the Document Type Definition (DTD) was developed to support the information element requirements, we chose to implement the authoring environment using Adept Editor/Publisher 9.0.1 on top of Astoria 3.4 (currently being upgraded to 3.6). There were two main reasons this solution was chosen:

- Valley Forge has implemented two very successful Adept/Astoria solutions for other customers in the past (Micro Compact Car (MCC) and Saturn)
- Many of the GPAS requirements were similar to MCC and Saturn, which allowed us to use existing code as a base.

Adept uses Formatting Output Specification Instance (FOSI's) and Adept Command Language (ACL) scripts to customize the authoring environment. In addition to the required DTD customizations, a few “hooks” were put into the environment to integrate the repository and authoring system a little closer. Some modifications were also made to the Adept/Astoria bridge. Below are some of the “hooks”:

- When the writer inserts a tag that represents a relational or “re-use” object, they are automatically presented with the Astoria “Select Object” dialog. This is the dialog that allows users to make data relational (reuse objects).
- A custom ‘Save’ button prompts users to enter extra revision information and assign a version status (Draft or Release)
- Hooks into the Section 2.4 and the Section 2.5 allow the writers to do all their work from a central interface.

The system also supports three different views of the data. Each one represents a different publication type. Writers can toggle between multiple views and publication types in the same SGML instance and authoring session, as shown in [Figure 2](#) and [Figure 3](#).

**Figure 2. PAD Screen FOSI View**

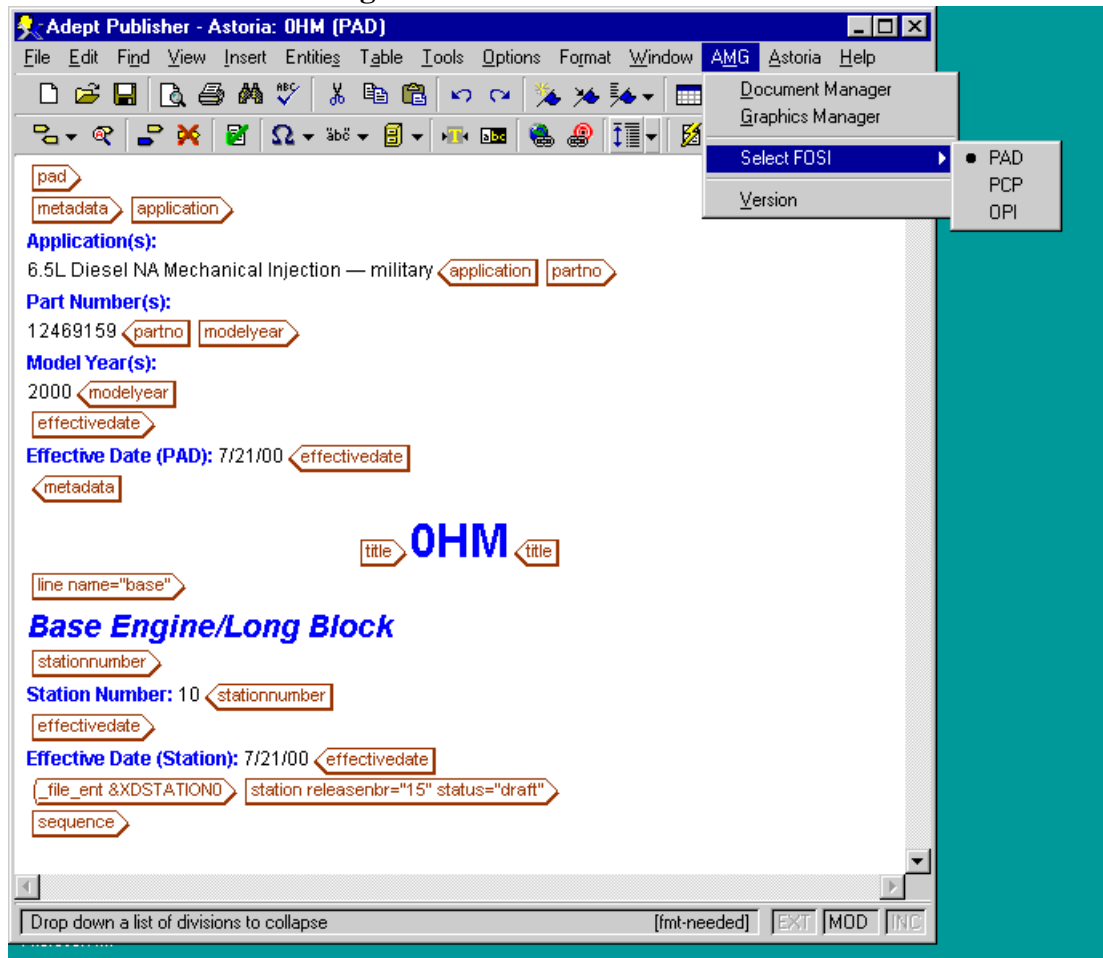
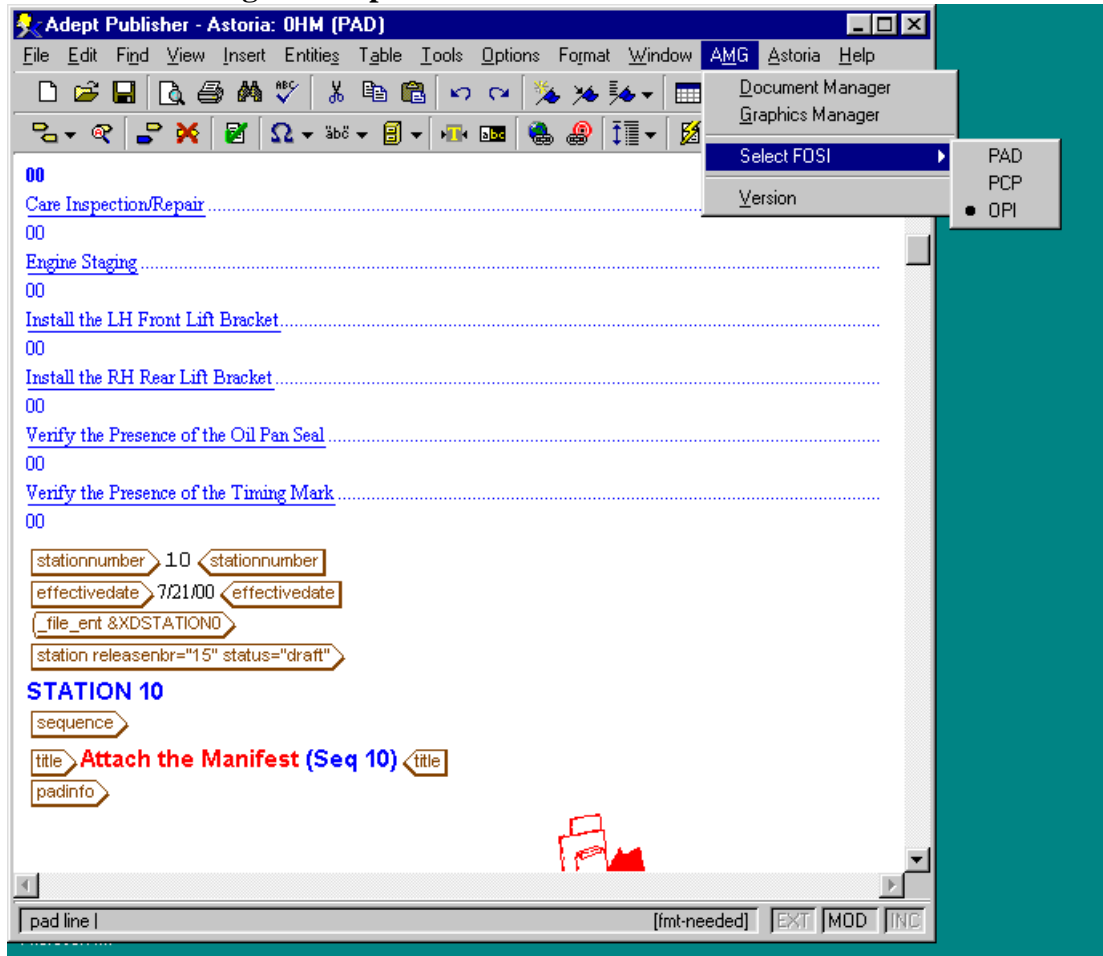


Figure 3. Operator Instructions Screen FOSI View



## 2.4. Document Manager

The document manager is a C++/Astoria Software Development Kit (SDK) SDK application designed to provide a simpler user interface than the Astoria Navigator. By simpler, I mean two things. First, it exposes only the features that the users need. Second, it automates and extends some features. Below is a list of some of the features the Document Manager provides:

- Copy/Move – Allows users to copy objects or move them to different folders in the repository. Users can choose between three types of copies: Normal, Relational, and Unique. (Figure 4)
- Create Child Folders – Allows users to create folders to help organize their data.
- Search/Replace – Allows users to perform global search/replace for part numbers while at the same time assigning a new version to each object affected. The GPAS system currently does not interface with the GEP parts database so this feature was necessary. Since tool names and numbers are stored as separate reuse objects, the writer can update the tool object and every place it is used will also be updated.



- Show reuse – Allows users to select an object and see each place that object is used. It also marks the instance that is the original object, as shown in [Figure 5](#).

**Figure 4. Copy/Move Screen**

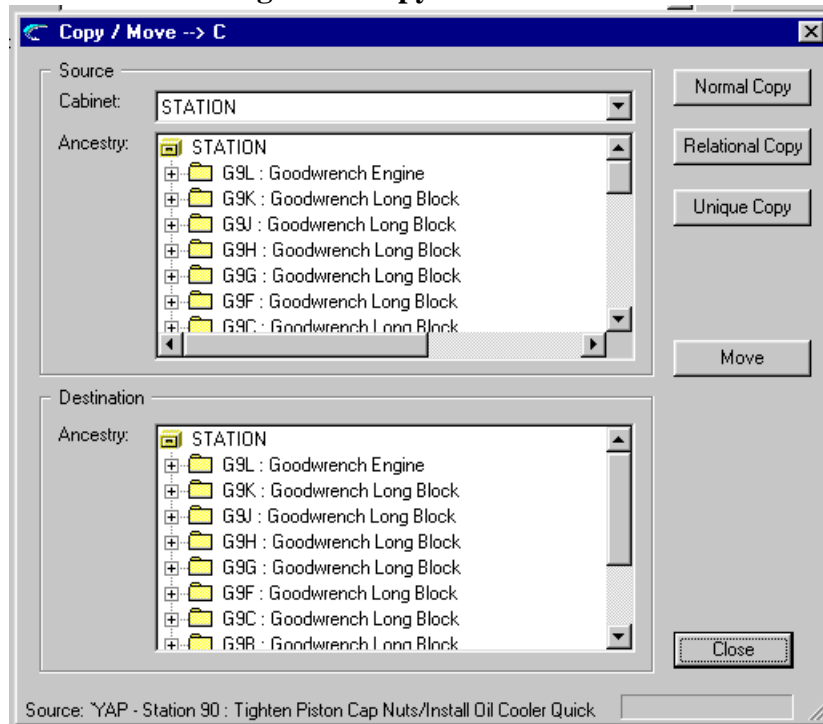
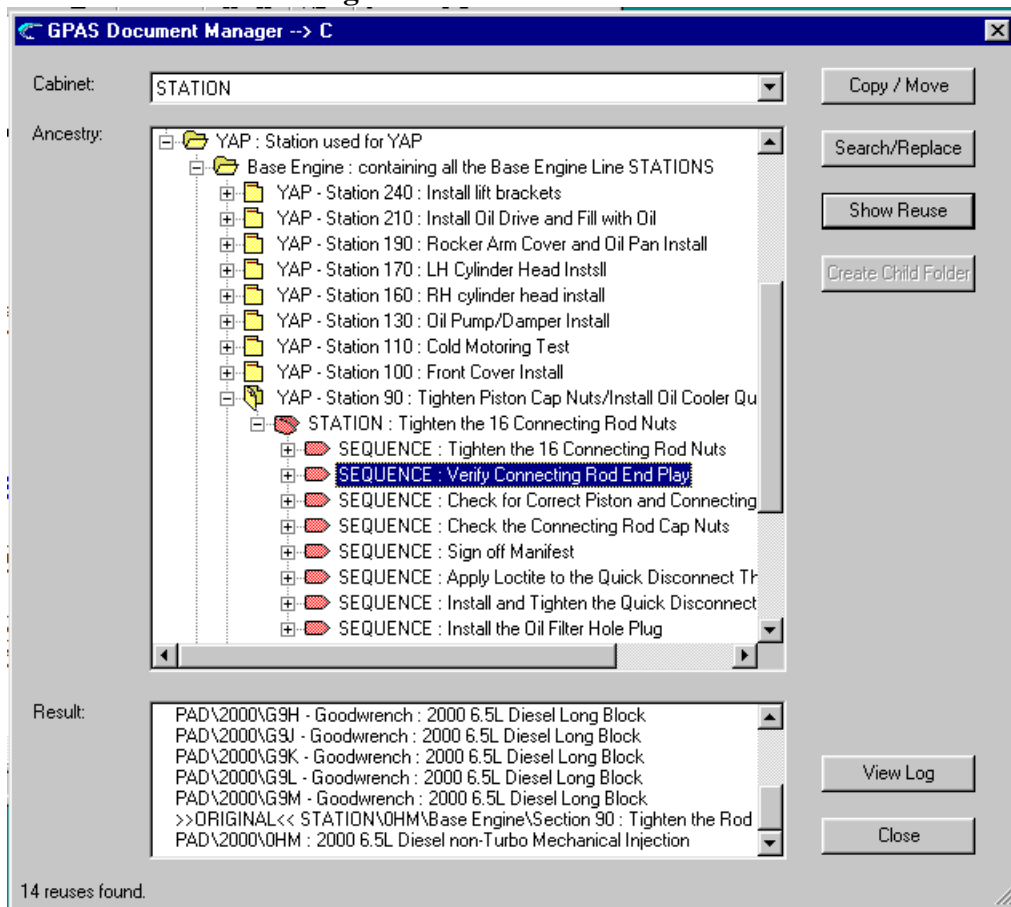


Figure 5. Show Reuse Results



## 2.5. Graphics Manager

The graphics manager is a C++/Astoria SDK application that facilitates the submission of graphics into the repository. One of the requirements was that the graphics, once put into the repository, would be easy to locate. This was accomplished in three ways:

1. If the user knows which PAD used the image they are looking for, a simple view of that PAD would indicate the name of the graphic used.
2. An “intelligent” naming schema was developed. The first two characters represent the OEM, the third represents the illustration category, such as engine block or intake manifold, and the remaining numbers are a sequential four-digit number.
3. Metadata is assigned to each image for searching. Figure 6 shows the sample metadata that is captured.

**Figure 6. Graphics Manager Metadata Input Screen**

The graphics manager prompts the user to select the illustration to be submitted into the repository, ensures the filename is correct, gives the user a preview of the image, prompts the user for the metadata, then automatically submits the illustration into the correct location in the repository.

## 2.6. Publisher

The publisher or composition engine is able to produce paper or Portable Document Format (PDF) output in three very different flavors. This is all done by a combination of FOSI's, ACL, and C++ code.

**Figure 7. Sample PAD Page**

OHM	Base Engine Long Block	Base-60-30-1				
Install the Cam Drive Gear Bolt and Washer						
	<p><i>Operation Description:</i></p> <ul style="list-style-type: none"> <li>Install the washer and tighten the bolt, then remove the locking fixture.</li> </ul>					
<i>Specifications:</i>						
Item	Part Number	Part Name	Quantity	Tool Number	Tool Name	Torque
1	14022548	Washer	1			
2	11503316	Bolt	1	TQWRENCH 02-811	Hand Torque Wrench Socket 18mm	160(170)180Nm
Issue Date: 7/21/00			Effective Date: 7/21/00			

Figure 8. Sample Operator Instructions Page

Base 60-27

**STATION 60**

Install the Injection Pump Drive/Timing Gear (Seq 10)

- Align the gear with the camshaft key and slide into position.

*Note: The injection pump drive timing gear is also marked with a dimple that will align with the crankshaft sprocket in the same manner as the camshaft sprocket.*

Item	Part No.	Part Name	Qty
1	14022653	Gear	1

Using the following tool(s):  
Compethane Hammer (Compethane Hammer)

Install Locking Tool In No. 1 Cylinder (Seq 20)

- Install the locking tool to lock the crankshaft into position.

Item	Part No.	Part Name	Qty
1	23402592	Crankshaft	0

Using the following tool(s):  
Plastic Tool (02-8917)

Install the Cam Drive Gear Bolt and Washer (Seq 30)

- Install the washer and tighten the bolt, then remove the locking fixture.

Item	Part No.	Part Name	Qty
1	14022648	Washer	1

Using the following tool(s):  
0

Item	Part No.	Part Name	Qty
2	11503316	Bolt	1

Using the following tool(s):  
Hand Torque Wrench (TQWRENCH)  
Socket 18mm (02-811)

Torque to the following Specification:  
160(17) 180Nm

Figure 9. Sample PCP Worksheet Page

<b>OHM</b>	<b>Base Engine/Long Block</b>	<b>Base-60-30-1</b>
<b>Install the Cam Drive Gear Bolt and Washer</b>		

**Special Instructions:**

- Install the washer and tighten the bolt, then remove the locking fixture.

**Specifications:**

Item	Part Number	Part Name	Quantity	Tool Number	Tool Name	Torque
1	14922648	Washer	1			
2	11503316	Bolt	1	TQWRENCH 02-811	Hand Torque Wrench Socket 18mm	160(170)180Nm

**PCP Worksheet:**

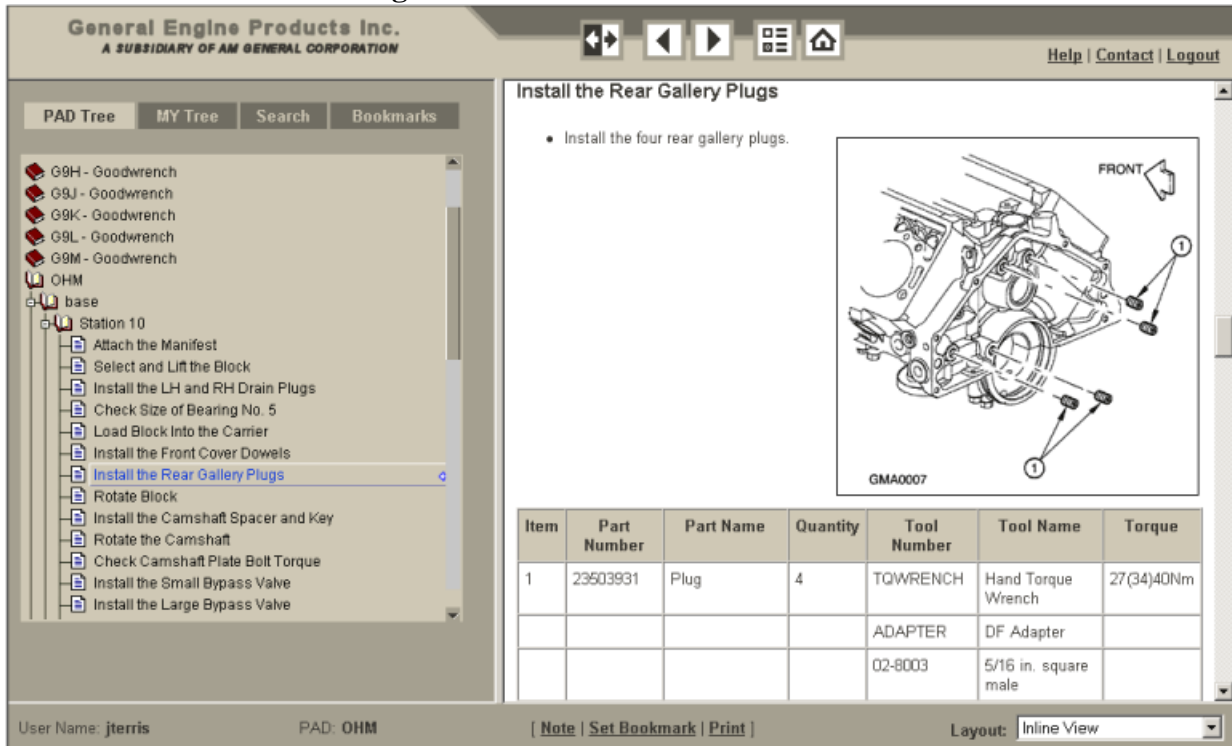
FAB   MOVE   STORE   INSP				Characteristics				Methods			
Sta. No.	Step No.	Process Name/ Operation Description	Machine, Device, Jig, Tools for MFG	Product	Process Spec. Chan. Class.	Product / Process Specification / Tolerance	Evaluation/Measurement Technique	Sample Size	Sample Freq.	Control Method	Reaction Plan
✓	60	30	Install the washer and tighten the bolt, then remove the locking fixture.	Hand Torque Wrench TQWR (ENCH), Socket 18mm 02-811							
	60	30	["Install / Place Requirements"]		Correct Part Used	Bolt (11503316)	Visual Inspection	100%	Continuous	Operator Training / Audit	Notify Supervisor, Tag Parts
	60	30	["Install / Place Requirements"]		Installed in Correct Location		Visual Inspection	100%	Continuous	Operator Training / Audit	Notify Supervisor, Tag Parts
	60	30	["Install / Place Requirements"]		Installed Correctly	Driven to Torque	Visual Inspection	100%	Continuous	Operator Training / Audit	Notify Supervisor, Tag Parts
	60	30	["Torque Requirements"]		Correct Torque	160(170)180Nm	Visual Inspection	100%	Continuous	Operator Training / Audit	Notify Supervisor, Tag Parts
	60	30	["Torque Requirements"]		Torque Correctly	Fully Driven to Torque; Not Cross-threaded	Visual Inspection	100%	Continuous	Operator Training / Audit	Notify Supervisor, Tag Parts

**Issue Date: 7/21/00**
**Effective Date: 7/21/00**

## 2.7. Web-Based Viewer

The GPAS Web-Based viewer is built using the Valley Forge Web Application Framework (VFWAF). Index information is stored in Oracle or SQL Server, while the content is pure XML. Using Java Server Pages (JSP) and XSL/XSLT, desired information is delivered to the client in Hypertext Markup Language (HTML). The viewer supports many navigation, search, presentation and user configuration features. Figure 10 is a sample screen shot that represents the typical tree navigation view.

Figure 10. GPAS Viewer Tree View



The left side is the navigation portion of the viewer and can be hidden by the user if desired. The right side is the content portion of the viewer that displays the content that represents the selected object in the navigation pane. The hierarchy of the PAD can be seen in Figure 10: PAD (e.g., OHM), line (e.g., base), station (e.g., Station 10), and sequence or step (e.g., Install the Rear Gallery Plugs). Along the bottom of the viewer, you will notice the following:

- User Name – Identifies the user who is logged into the system. Each user is allowed to change user-specific settings such as default layout, user interface language, and content language. User-specific notes and bookmarks are also supported.
- PAD – Identifies the current PAD represented in the content pane.
- Note and Set Bookmark – Allows users to set content-specific notes and bookmarks.
- Print – Generates a printer-friendly version using a special XSL.
- Layout – Allows the user to switch the layout or presentation style. The current layouts supported are Inline view, similar to the PAD format; Operator Instructions view (Figure 11); Text Only, for low bandwidth (Figure 12); and Expert view, which shows only the image and specification table (Figure 13).

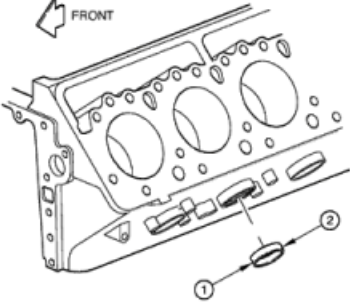
Figure 11. Operator Instructions View

General Engine Products Inc.  
A SUBSIDIARY OF AM GENERAL CORPORATION

Help | Contact | Logout

### Station 50

#### Install the Block Core Plug (Seq 10)



GMA0023

- Apply Loctite no. 15046D to the block core hole.
- Drive in the block core hole plug.
- Wipe off the excess Loctite.

Note:  
The plug must be flush or below the chamfer.

User Name: jterris      PAD: OHM      [ Note | Set Bookmark | Print ]      Layout: Operator Instruction View

Figure 12. Text Only View

General Engine Products Inc.  
A SUBSIDIARY OF AM GENERAL CORPORATION

Help | Contact | Logout

Station 50-Seq 10

#### Install the Block Core Plug

[ GRAPHIC GMA0023A.GIF ]

- Apply Loctite no. 15046D to the block core hole.
- Drive in the block core hole plug.
- Wipe off the excess Loctite.

Note:  
The plug must be flush or below the chamfer.

Item	Part Number	Part Name	Quantity	Tool Number	Tool Name	Torque
1	9985345	Loctite	1	BRUSH	Brush	
2	838538	Plug	1	02-8673-001	Air Drift Tool	
				02-8670	Drift Tool	
				51-6007	Hammer	

Station 50-Seq 20

#### Perform TNT and Inspection

User Name: jterris      PAD: OHM      [ Note | Set Bookmark | Print ]      Layout: Text only View

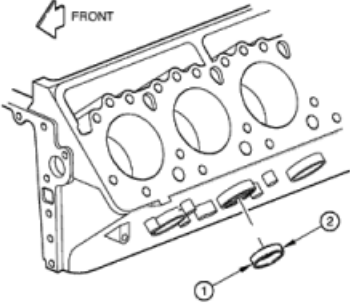
**Figure 13. Expert View**

General Engine Products Inc.  
A SUBSIDIARY OF AM GENERAL CORPORATION

Help | Contact | Logout

Station 50-Seq 10

Install the Block Core Plug



GMA0023

Item	Part Number	Part Name	Quantity	Tool Number	Tool Name	Torque
1	9985345	Loctite	1	BRUSH	Brush	
2	838538	Plug	1	02-8673-001	Air Drift Tool	
				02-8670	Drift Tool	

User Name: jterris      PAD: OHM      [ Note | Set Bookmark | Print ]      Layout: Expert View

Currently the viewer supports three types of searches. The first is a text search. Since the data is stored in XML, users can search for specific text located in document titles only or anywhere in the documents. The second is a part number search. This is shown in Figure 14.

**Figure 14. Part Number Search**

General Engine Products Inc.  
A SUBSIDIARY OF AM GENERAL CORPORATION

Help | Contact | Logout

PAD Tree   MY Tree   Search   Bookmarks

Search for Text:

Document Titles    Full Text Search

Search for Part Nr.:

Search for  
PAD Nr.:    Line:    Station Nr.:



User Name: jterris      [ Note | Set Bookmark | Print ]      Layout:



The results of each search are presented to the user in the navigation pane, as show in [Figure 15](#).

**Figure 15. Search Results Page**

The screenshot displays the search results page for General Engine Products Inc. The left navigation pane lists search results for 'Install the Bypass Cup Plug' and 'Verify Presence of Large Bypass Valve Cup Plug'. The main content area shows the detailed view for 'Verify Presence of Large Bypass Valve Cup Plug'. It includes a diagram of the valve assembly with a callout for the cup plug and a table of tools required for the task.

Item	Part Number	Part Name	Quantity	Tool Number	Tool Name	Torque
1	14066310	Bypass Cup Plug	0	Crayon	Crayon	

The last search type is by [PAD](#), line, and station. This is the search mechanism that allows the barcode reader to deliver context-specific data to the user based on the manifest.

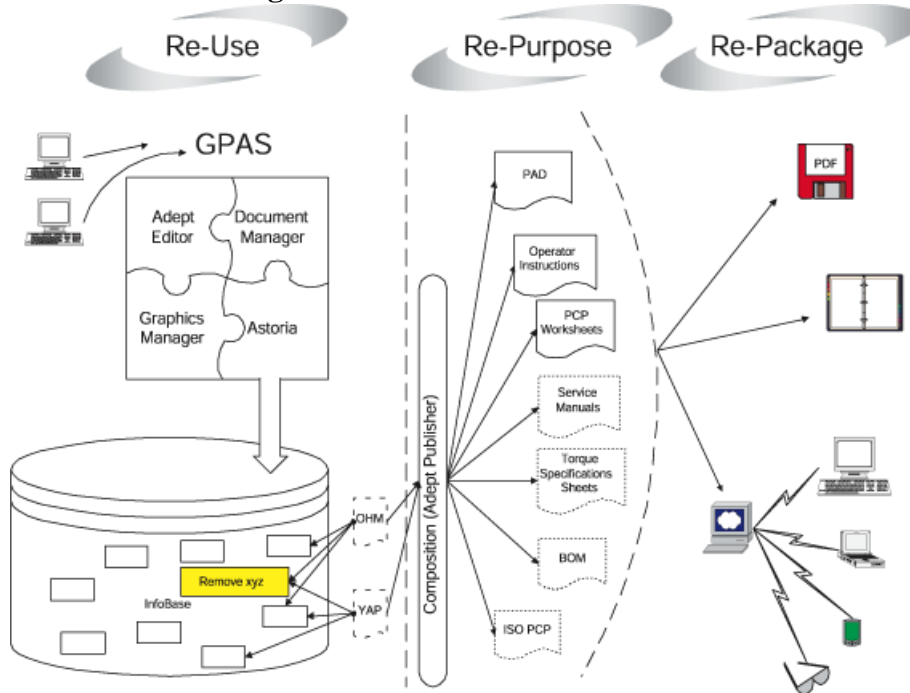
At this time, the viewer has not been deployed to the plant floor. The final delivery system has not been defined. However, the [GEP PAD](#) administration team has found the viewer to be very helpful. The next section takes a look at [Section 3.2](#), [Section 3.3](#), and [Section 3.4](#) in more detail and describes some of the implementation strategies used in [GPAS](#).

### 3. Re-use, Re-purpose, Re-package Under the Hood

#### 3.1. Overview

Before we dive into these topics, we need to define what is meant by these terms. Re-use means to use the EXACT same information object (image or text) in more than one “document”, view, or content object. Re-purpose means to extract or format the same piece of information in many different ways, usually producing a different document type, view, or presentation. Re-package means to deliver each document type, view, or presentation via different media and/or delivery systems. [Figure 16](#) gives an overview of how these three concepts relate to [GPAS](#).

**Figure 16. R3 Overview in GPAS**



### 3.2. Re-Use

Re-use has been around for many years. Re-use can be implemented simply as external image references or in a more complex manner such as Object Oriented database repositories, where each piece of text is stored only once. A common term used in conjunction with re-use is granularity. Granularity refers to levels in the SGML/XML hierarchy. The [GPAS DTD](#) supports the following hierarchical structure:

```
<pad>
  <line>
    <station>
      <sequence>
      </sequence>
    </station>
  </line>
</pad>
```

This represents four levels of granularity; pad, line, station, and sequence. Assuming images and other elements are allowed somewhere in the <sequence> level, one could say there were five levels. The important thing to remember with re-use is the lower the level of granularity you support for reuse, the more expensive it is in terms of support, system maintenance, repository strength, and in most cases performance. However, the more re-use you can build into your system, the more savings you can realize in the costs of translation, content development, content maintenance (which usually takes a much higher skilled content developer), and in reduced content errors. The key is to find the right trade-off and not to over-engineer the repository.

GPAS allows re-use at the station, sequence (step), tool, and image level. This is the reason there are three different copy features on the document manager. Let's assume the following PAD exists in the repository:

```
<pad>YAP
  <line>base
    <station id=st1>content
      <sequence id=sq1>content
      <sequence id=sq2>content
    </station>
    <station id=st2>content
      <sequence idref=sq1>pointer to sq1
      <sequence id=sq3>content
    </station>
  </line>
</pad>
```

Images and tools are always re-use objects so no matter what type of copy is performed, they will always remain re-use objects.

**Table 1.**

Results of a Unique Copy	Results of a Relational Copy	Results of a Normal Copy
<pre>&lt;pad&gt;YAP2   &lt;line&gt;base     &lt;station id=st3&gt;con-     tent       &lt;sequence id=sq4&gt;con-       tent       &lt;sequence id=sq5&gt;con-       tent     &lt;/station&gt;     &lt;station id=st4&gt;con-     tent       &lt;sequence id=sq6&gt;con-       tent       &lt;sequence id=sq7&gt;con-       tent     &lt;/station&gt;   &lt;/line&gt; &lt;/pad&gt;</pre>	<pre>&lt;pad&gt;YAP3   &lt;line&gt;base     &lt;station     refid=st1&gt;pointer to st1     &lt;/station&gt;     &lt;station     idref=st2&gt;pointer to st2     &lt;/station&gt;   &lt;/line&gt; &lt;/pad&gt;</pre>	<pre>&lt;pad&gt;YAP4   &lt;line&gt;base     &lt;station id=st3&gt;con-     tent       &lt;sequence       refid=sq1&gt;pointer to sq1       &lt;sequence       refid=sq2&gt;pointer to sq2     &lt;/station&gt;     &lt;station id=st4&gt;con-     tent       &lt;sequence       refid=sq1&gt;pointer to sq1       &lt;sequence       refid=sq3&gt;pointer to sq3     &lt;/station&gt;   &lt;/line&gt; &lt;/pad&gt;</pre>

Remember, another form of re-use is reusing something as a template. Who out there develops code, proposals, procedures, etc. starting with a blank page?

### 3.3. Re-Purpose

Re-purposing is just starting to become more than simple format changes. One of the major benefits of SGML and XML is the separation of content and presentation/format. A few years ago, as long as a publishing system could compose the same instance in at least two ways (e.g. single column and two-column) we said the system supported re-purposing. Now the re-purpose standard is higher. The challenge is to develop content management systems and standards that enable content developers to create information for many purposes, not formats, at the same time. The good systems and content development standards do not even know how the content

will be used. It might appear in a marketing brochure, a service procedure, an installation guide, a training manual, an assembly procedure, and an owner's manual or any combination thereof.

**GPAS** currently supports the following three publication types: **PAD**, Operator Instructions, and Process Control Worksheets. The content development process and **DTD** were designed to support the following additional publications:

- Service Manuals
- Bill of Materials (parts lists)
- Tool Handbooks
- Torque Specification Sheets

### **3.4. Re-Package**

XML and its family of standards have really enabled this feature. Typical target media include paper, Web Server (electronic), and CD-ROM/DVD. Platforms are becoming more diverse and, in some cases, impact the composition engine very much. They include desktops, laptops, PDA's, Palm PCs, tablet computers and other handheld devices, hands-free devices, and even phones. The way you present information on a 21-inch desktop with unlimited bandwidth is very different from the way you present data on a phone or PDA.

The **GPAS** system can deliver information in paper format, PDF, and via the **GPAS** Web viewer. The viewer is able to detect device-specific information, which allows it to customize the view based on the browser version, Operating System, and display context.

### **3.5. Implementation Techniques**

The following are a few implementation techniques that can be used to enable re-use, re-purposing, and/or re-packaging in your system. Some of these techniques were used in **GPAS**.

#### **3.5.1. White Text**

This technique, in its literal form, displays or prints the text using the font color that matches the background. By doing this, the text is hidden. This works great for tests or forms. For tests by displaying both the questions and answers in the default color, you have yourself an instructor's answer key. Display the questions in the default color and answers in the background color and you have the test itself. Depending on what type of information you design into the form, you could make a blank form, a sample form, or even a training guide from the same instance. This technique does not work well when pagination is dependent on the existence of text.

### 3.5.2. Profiling

In profiling, the style and even content of a displayed document is changed based on the value of elements and/or attributes and the profile selected. This technique can be applied to both SGML/XML elements and attributes, or to the combination of the two.

In the element implementation, the composition engine is developed to use a subset of the tags depending on the purpose. Entire constructs or hierarchical levels may be included or excluded. Elements that are used may also require different processing depending on the purpose.

In the attribute implementation, usually one common attribute is added to every element. The value of the attribute determines how and when the content should be used. Profiling this way can be used to identify the following:

- Effectivity – What specific configuration (feature set, language, market) must be satisfied before this content applies.
- Delivery Platform – There are many things you can do in an electronic presentation that you cannot do in paper. There are also many things you can do with computer-based diagnostic equipment that you may not be able to do manually.
- Purpose – If the purposes (publication types) are finite and defined, this may be the most effective way of denoting what content goes with what.
- User skill or knowledge level – The information you give users with 20 years of experience may be different from what you give beginners.

### 3.5.3. Late-Link Binding

I have found this technique to be the single most powerful technique I have ever used. I first implemented it in 1991 and have used it ever since. The easiest way to describe it is with a very common example: Domain Name Server (DNS). DNS resolution is one implementation of late-link binding. When you enter the Universal Resource Locator (URL) <http://www.vftis.com>, somewhere that URL is mapped at run-time to an IP address. As long as the DNS mapping between URL name and number are up-to-date, the IP address associated with <http://www.vftis.com> can be changed with no impact on the end user.

With late-link binding, there can be many levels of indirection. The following table illustrates this idea.

**Table 2.**

Level of Indirection	1	2	3	4
	IP address identifies a physical machine	URL	Value in your local host table	Query or search on Yahoo

With each level of indirection you risk being directed to multiple destinations. In some cases that is OK, but in the ideal world you design the system and content to resolve to exactly where the user wants to go.

What is so special about late-link binding? With it you are able to maximize re-use and re-purposing because the content does not have to be re-created just because an image is modified or the external reference is different due to the content's context. Lesson numbers, section numbers, steps and items in ordered lists do not have to be rewritten just because they are first in one context and last in another. Yes, auto-number features in your composition engines are another form of late-link binding.

One late-link binding technique involves implementing a "smart" referencing ID schema. I sometimes call this a "promise ID". This technique facilitates late-link binding, and also allows content that is dependent on other content to be developed concurrently, including the inclusion of references to content that is not yet developed. By having the "promise" that the content you need is at a predefined address, you can include your reference even if it has not been created yet. Here are some tips in developing such a schema:

- Define the level of granularity to be addressed up front.
- If possible, develop a master outline that contains all possible hierarchical nodes or branches.
- Assign unique IDs to each value in the hierarchy.
- Create the referencing ID by concatenating the ID's from each level in the hierarchy.
- Find some way to separate or denote each level of the hierarchy within the schema. This will pay big dividends later in case a referencing error handler is necessary. I will let the reader think about that.

The ultimate late-link binding schema in my mind is one that implements HyTime addressing schemas such as queryloc. In this implementation, instead of referencing graphic IDs, writers encode queries based on some standard such as XPointer, XQL, or SQL, or some custom language better suited for the writer. As with all late-link binding implementations, the late-link binding engine interprets the addressing schema, applies current context information, and resolves to the correct information object.

### **3.6. When in the Process**

Re-Use, Re-Purpose, and Re-Package techniques are performed at three different points in the delivery cycle. First there are Request-Time Transformations. These transformations are done on the server side only and usually require some sort of XSLT, COM, Java, Perl, Omnimark, or ASP/JSP process to be done prior to delivering content to the user. The results of the transformations usually depend on one or more factors such as user preferences (skill level, knowledge level, language, etc.), user selections, current context, and client browser. If currency of the data is the most important factor and your system supports a very high level of reuse, repurpose, and repackaging, your delivery system will need to implement these transformations.

Next there are Run-Time Transformations. These transformations are limited to presentation styles and some late-link binding referencing resolution. They do not provide the power of the Request-Time transformations but are much faster. The transformations can be done by server-and/or client-side XSL, CSS, or DHTML. They can also be implemented by custom browser extensions using ActiveX, Java, COM, etc. When people describe solution that support “style-sheet switching”, they are typically describing these type of transformations.

Finally there are Pre-Delivery Transformations. This implementation provides the fastest solution possible. However, it requires that all possible variants of the data be defined prior to delivery. Because each of these variants is prepared in advance as a separate file, this method requires more space on the server than the other methods. Most often, you implement Request-Time transformations for a defined set of variants prior to publishing on the delivery system.

### **3.7. Emerging “Enabling” Technologies**

As [GEP](#) vision of reuse, re-purposing, and re-packaging is developed and implemented, new technologies also mature and can be utilized. Some of these are described below.

#### **3.7.1. SVG**

[SVG](#) brings XML to vector graphics. The fact that [SVG](#) is expressed in XML means that [SVG](#) graphics can be just as dynamic as XML text. Also, the fact that it supports scripting increases the flexibility.

#### **3.7.2. XMP**

[XMP](#) has been developed by Adobe Inc. to provide Adobe applications and partners with a common, Standards-based metadata framework that standardizes the creation, processing and interchange of document metadata. The [XMP](#) software development kit allows programmers to integrate XML metadata packets within application files. These packets can be used in metadata exchange and resource discovery, creating opportunities for digital rights management, job processing, workflow automation, and other areas of production where metadata is critical.

### **3.8. XSL-FO**

XSL-FO is an XML-based markup language that lets you specify the pagination, layout, and styling information that will be applied to your content. XSL-FO markup is complex and verbose, and there are currently few tools to process it effectively. For these reasons, XML-FO has been slow to catch on. Will we ever get to the point where one “style-sheet” will really support multiple output formats? Will XSL-FO be the technology that gets us there?

## 4. Conclusion

Analysis of **GEP** business needs helped define a more efficient documentation system, where information could be written once the re-used for multiple engines, re-purposed in multiple manual types, and re-packed for several delivery platforms. Applying a known technology, SGML/XML, in an existing industry has been fruitful. As new technologies appear, their applicability to the problem of document distribution can be evaluated.

**GEP** vision is to have a system where both manufacturing and product engineers could develop and maintain the information they are responsible for while reducing the duplicate effort, increasing the overall quality of the information, and minimizing publishing latency. The techniques outlined in this paper are helping **GEP** get closer to this vision.

## Acknowledgements

First of all I would like to thank Jeff Dowell for his never-ending support and confidence in Valley Forge and me. Without that, this project would not be where it is today. I would also like to thank my mentor Denise Feil who taught me everything I know about SGML/XML and publishing. I would also like to thank my colleague Janet Erickson. Without her endless technical and non-technical edits, this paper would not be fit for publishing.

Finally, I want to especially thank my family for all their support and love they have given me over the years. Tracy, Austin, and Kyleigh, I love you.

## Glossary

ACL	Adept Command Language
DNS	Domain Name Server
DTD	Document Type Definition
FOSI's	Formatting Output Specification Instance
GEP	General Engine Products, Inc.
GM	General Motors
GPAS	GEP PAD Authoring System
HTML	Hypertext Markup Language
ISO	International Standards Organization
JSP	Java Server Pages
MCC	Micro Compact Car



OEM	Original Equipment Manufacture
PAD	Product Assembly Document
PCP	Process Control Plan
PDF	Portable Document Format
SDK	Software Development Kit
SVG	Scalable Vector Graphics
UPC	Universal Product Codes
URL	Universal Resource Locator
VFWAF	Valley Forge Web Application Framework
XMP	Extensible Metadata Platform
XSL-FO	XSL Formatting Objects

## Biography

### John F. Terris

IT Program Manager, Allen Park  
 Valley Forge Technical Information Services  
 Allen Park  
 U.S.A.  
 Email: [john.terris@vftis.spx.com](mailto:john.terris@vftis.spx.com)

*John F. Terris* graduated from Eastern Michigan University in 1991 with a B.S. in Computer Science. After graduation, he was employed by Ford Motor Company and worked for the Technical Publication Systems group. At Ford, he was involved in the development of many DTD's, SGML conversion projects and was the Project Manager of the Ford of Europe Technical Information System. He was also active in the J2008 DTD committee and was a speaker at the 1994 CALS EXPO.

Terris is currently the Program Manager of Information Technologies for SPX Valley Forge Technical Information Services in the Allen Park office, where he is able to continue his passion for developing SGML/XML Authoring, Publishing and Delivery systems for clients such as Ford Motor Company Technical Training, General Engine Products, Harley-Davidson, Hyundai, and others. His proudest accomplishments are his wonderful marriage of 10 years to his lovely wife Tracy and his two children Austin and Kyleigh.