ON THE ROLE OF "REUSABLE OBJECTS" IN THE EVOLUTION OF VARIABLE DATA PRINTING AND INTEGRATED CROSS-MEDIA PUBLISHING

JACOB AIZIKOWITZ, PH.D.
PRESIDENT, XMPIE

9/11/2009

Abstract

This paper examines the unique role that Object-Based Variable-Data Page Description Languages (OB vPDLs) played in the evolution of color digital printing. Personalization in printed documents — known as Variable Data Printing (VDP) — is one of the killer apps for color digital printing; however, without a page description language that could address the unique requirements of a VDP job, this app could not flourish.

OB vPDLs were invented — in the mid to late 90's — in order to fill this void in vPDLs. These languages were built on the foundations of the desktop publishing revolution, which had Adobe Postscript as its leading PDL, and augmented it with language constructs that allowed for fast rendering, by the RIP, of series of individualized pages that were individualized and creative. This style of VDP opened the door for the use of print personalization outside of just transactional printing, and it made VDP a mainstream choice in 1:1 Marketing. Also, the object-based nature of the OB vPDLs opened the door for integrated cross media solutions.

Table of Contents

ON THE ROLE OF "REUSABLE OBJECTS" IN THE EVOLUTION OF VARIABLE DATA PRINTING AND INTEGRATED CROSS-MEDIA PUBLISHING

HOW VPDLS WITH "REUSABLE OBJECTS" CATAPULTED VARIABLE DATA PRINTINGING TOBECOME A MAINSTREAM SOLUTION IN MARKETING AND DRIFT AWAY FROM TRANSACTIONAL… AND OPENED THE DOOR FOR INTEGRATED CROSS MEDIA PERSONALIZATION

INTRODUCTION

The evolution of color digital printing since about the mid '90s brought the promise of using highly creative and personalized print materials in direct mail and other print-based applications. Regretfully, at that time, the protocols for communicating variable data printing-jobs to printers were typically forms-oriented and had no power to express the creative variability that this new breed of color digital print engines enabled. This void prevented the materialization of the color digital printing promise, and it's the closing of this void that Object-based Variable-data Page Description Languages (OB vPDLs) were invented for.

OB vPDLs were novel in defining a page model that was built out of a mix of *unique elements* (typically text), and *reusable elements* (typically graphics or images, but also text snippets). This was in contrast to the approach that formalized the page background — like a Form — as the only reusable element in a job (and usually named it Master Page); this forms-based approach to VDP also restricted the variability in a page to the areas that represented the fields of the form.

Key novelties of the OB vPDLs were (a) that the elements themselves — unique or reusable — were defined using Postscript (or, later, PDF), and (b) that there were very little restrictions on (1) what components of a page can be elements, (2) how elements can be placed on the "page plane", (3) whether elements can obstruct each other, or (4) how many elements can be used in a page.

This unconstrained object model made OB vPDLs capable of specifying pages and jobs that were almost unlimited in terms of their graphical richness, yet, because of the reusability of objects allowed for rendering of pages at speeds that met the needs of the color digital printing engines.

Having such expressive[1] vPDLs opened the door to the development of applications for creating variable data documents that made no restrictions on the user's creative freedom while authoring or designing such

[1] Expressive here means that the language could describe pages with practically no limitations on the type of graphical elements — text, images, etc. — that appear on them or the interaction of such elements, such as partial or full obstruction, etc., etc.

documents. This is because no matter what the designer would want to create, its reproduction in variable data print was possible.

So, OB vPDLs became the bridge that connected creative and speedy rendering for high-quality, productive, VDP.

In the below we will examine this evolution in more details, and we will also present key insights on implementation.

THE ROLE OF OB VPDLS IN MAKING COLOR VARIABLE DATA PRINTING A PUBLISHING DISCIPLINE AND NOT JUST AN I/T TOOL

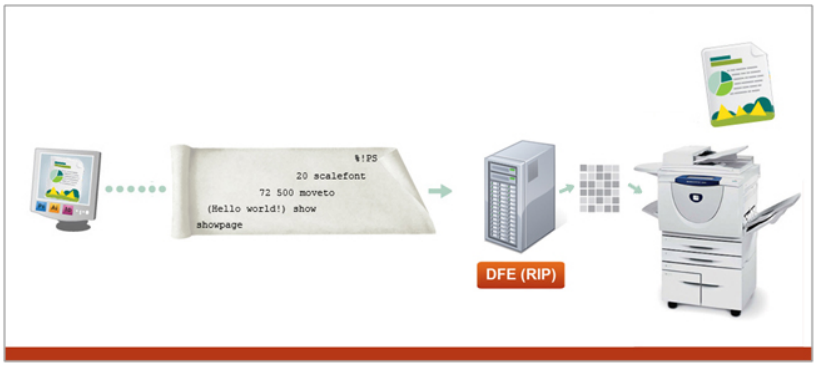

FIGURE 1: DTP WORKFLOW

Our main thesis in this paper is that OB vPDLs did not simply expand the use of forms-based workflow to color. Instead, OB vPDLs revolutionized VDP by taking the well known and proven workflow of desktop publishing (DTP) and expanding it into the VDP space. This opened VDP to the creative community and made highly creative VDP a reality.

Below we will introduce the essentials of desktop publishing (DTP) and how they are related to enabling creative. We will then show how OB vPDLs enabled bridging *creative* and *print-efficiencies* and, by that, opened the door for DTP in the VDP world.

The Foundations of the Desktop Publishing (DTP) Workflow

DTP workflow starts with creating a document, on the desktop, using software such as Adobe InDesign. For printing, the software generates a stream of instructions in a PDL — typically Adobe PostScript — and sends this stream to the print engine[2]. This stream is processed by a controller — Digital Front-End (DFE) or RIP[3] — which creates a series of page-bitmaps that it feeds to the print engine, resulting in a series of printed pages. The key property of this workflow is that what one sees on the desktop's screen is what one will see on the printed page. Hence creation-to-reproduction fidelity is easy to achieve and control by the originator — designer, author, etc. Postscript was a critical enabler of this property because:

1. Any page — simple or graphically-rich — can be specified using PostScript[4].
2. Hence, any document created interactively on the desktop can be faithfully translated to PostScript for printing.
3. Any page specified in Postscript can be faithfully rendered in print,
4. Hence, any desktop-created document can be faithfully rendered in printed form.

[2] It is commonly accepted that the desktop publishing revolution happened in 1984 when Aldus PageMaker, Adobe Postscript, and Apple's LaserWriter all came together.
[3] RIP stands for Raster Image Processor and it's a common name for the process that converts a stream of print instruction (in a PDL) to a raster — a page bit map or, more precisely, a page pixel-map (where a pixel may be represented by many bits, representing the various colors and the gray-scale level of each color component).
[4] PostScript is a full programming language around a full range of graphic operators, hence it can specify anything one needs

With such powerful PDL the concept of What You See Is What You'll Get (WYSIWYG) emerged and, in essence, was at the heart of the DTP revolution — control moved from the few print-room craftsmen to the **many** designers and originators of content.

Without applying WYSIWYG to VDP, the world of VDP would have stayed at the hands of the few print/data craftsmen, as was publishing before the days of DTP.

WYSIWYG for VDP is Challenging

Providing unconstrained WYSIWYG for VDP, which is crucial for being able to author creatively compelling individualized documents, is not so easy. A simple solution would be to use PostScript for specifying each page and each document individually. It will provide the WYSIWYG property, but it will make the RIP process too slow. This is because with "individual PostScript per page" the RIP cannot reuse the work done in interpreting one page for similar work needed for the next page. As a result, RIP-time would be unacceptably long, not keeping with the speed requirements of digital print engines.

Solving the RIP-speed issue without sacrificing the graphical and textual richness of the vPDL[5] became the major challenge for the industry. Object Based vPDLs were the solution.

Object-Based languages

Careful analysis of variable data print jobs revealed the key insight behind object-based VDP languages: *for any given VDP job, the individual pages are not arbitrarily different from each other. They share many objects that are common for the job. This commonality, if properly specified in the print stream, can be leveraged for gaining RIP-speed by avoiding unnecessary reprocessing.*

Envision a job of 10,000 pages, each with a logo image and other stuff. Four different logos are used for the entire job. Each page has its own logo, and it may differ from the one used in other pages. Since there are only four logos, many pages will share the same logo and one could gain a lot, performance-wise, by re-using a rendered logo image after it was processed (RIP'ed) the first time.

FIGURE 2 CONCEPTUAL VIEW OF VDP

The understanding that objects can be used to describe the commonality across pages of a VDP job drove the creation of Object-based VDP PDLs (OB vPDLs)[6]. Rather than constraining the page structure (i.e., using Forms), the idea was to constrain the vPDL to explicitly specify objects, and to specify pages as a mixture of page-unique content and previously defined "re-usable" objects. With OB vPDLs, processing-speed gains would come from being able to reuse previously processed objects rather than from repeating a constrained page structure.

[5] Forms-based vPDL gave RIP-speed gains but forced all pages to look like forms with filled-in data, which was certainly a very significant disabler for creativity.

[6] The authors, when employed by Scitex Corp, created one of the first object-based languages (1997) which was called VPS and was based on Postscript; the VPS language, known as Creo VPS™ at the time of this writing, was a critical foundation in

Examining the RIP's workflow will help realizing where speed-gains come from:

(a) *For each new object, process the object's definition and store the resulting raster-image (bitmap) in the objects' cache; and*
(b) *For each page in the job, use such cached objects as part of the construction process for the page bitmap.*

Typically, the number of different elements in the objects' cache would be much smaller than the number of pages in a job. Hence, for most of the pages, aside for rendering the page-unique stuff, the RIP will just retrieve from the cache bitmaps of previously rendered objects and merge them into the page bitmap. And this is where RIP speed-gains come from.

IMPLEMENTING OB VDP: FROM AUTHORING DYNAMIC DOCUMENTS TO GENERATING THE OB VPDL FOR PRINTING THEM

There are two challenges in the authoring of dynamic documents:

1. **Document design**: how to design documents that are both creative and dynamic; and
2. **Document production**: how to convert a dynamic document and a list of recipients to an OB vPDL output stream

Document Design

There are numerous ways to handle the document design challenge, yet today's designer-focused solutions are all based on augmenting existing publishing software applications (e.g., Adobe InDesign) with tools that add variability. Designer-focused solutions preserve the desktop publishing software's native document structure and add variability by tagging document objects that need to be dynamic with information that allows for changing such objects to reflect recipient-specific data[7].

For example, a dynamic graphic-frame will have a tag that indicates which picture to choose for populating the frame; such indication may be based, for example, on the car type that the recipient prefers[8].

Regretfully, at the time of this writing, OB vPDLs still have few graphic-model gaps relative to what one can create with desktop publishing design software or with what one can specify in Postscript. These gaps mean that a graphic capability that is supported by the design application may be impossible to describe (specify) in an OB vPDL and, hence, impossible to render in print. One notable example is the lack of support for transparency between overlapping objects, where transparency is fully supported by the design application.

Obviously, in areas where gaps exist, one cannot provide true WYSIWYG, and the printed document will appear different than the document on the screen. Hence, in order to maintain WYSIWYG, the document design tools should prevent the user from creating effects that cannot be faithfully rendered in print, or at least warn them that there is an issue.

It is expected that many of the graphic model gaps will be eliminated with the introduction of PDF/VT from Adobe.

---

the creation of PPML which provided similar concepts but in an XML framework, allowing more flexibility between Producers and Consumers. Between the years 2000 and 2004, the only practically-viable object-based VDP language was VPS and its implementation—on the consuming side—by the Creo Controller.

[7] Such solutions will avoid converting the publishing software's native document format to some proprietary internal format because by doing this they create WYSIWYG fidelity challenge and workflow complexity for handling changes.

[8] It is highly desired that the various graphic properties of such a frame — including the policy of content and frame fitting — must be adhered-to repeatedly each time the frame is populated with new personalized content.

Document Production

Generating the OB vPDL output stream for a given dynamic document and a given list of recipients is accomplished using a process that scans the recipients' list, and, for each recipient, does:

1. Creates the content for all needed dynamic objects (typically by interacting with database information and computing rules that were defined ahead of time),
2. Populates the dynamic objects with the content created for them,
3. Applies reflow and other layout adaptations that are needed in order to accommodate the recipient-specific content, and
4. Creates the section in the OB VDPL stream that will cause the printing of the document instance for that recipient.

One of the biggest challenges in generating the OB vPDL output stream is identifying the parts of the document that can become reusable objects. The heart of a good output- generator (also known as "composition software") is analysis of the document structure so that the elements of the document that can become reusable objects of the output stream will be identified. Once identified, another critical component of the output-generation algorithm is the interaction with the document layout technology (e.g., in the world of XMPie, Adobe InDesign Server or XMPie XLIM Server) in order to render a reusable element into an object and then store it in the objects dictionary[9]. Finally, the algorithm needs to identify a repeat use of such an element and use the dictionary to fetch a reference to its object rather than re-create the object.

Also critical is the decision whether an element in the document should be defined as a reusable object or treated as a unique element (known as "inline" in some dialects of OB VDPLs). Unique elements are being rendered into the output stream and there is no need to remember them or be able to later reference them. Failing to identify an element as reusable and, instead, classifying it as "unique" results in an output stream that would still render the document faithfully but will be much less efficient, which, at the extreme, as slow as "unique Postscript per page", which may severely impact the ability to produce the job efficiently.

Identifying the elements that can become reusable objects and how best to place them in the output stream is where different document creation and composition packages may vary significantly; and this is also an area rich with issued patents, pending patents, and just unique technologies.

Finally, the ability to maintain an Objects Dictionary across job boundaries can be a significant advantage for high-speed processing of repeat jobs. Once the output-generation for the first job in a series created the Objects Dictionary, output generation for subsequent jobs will be able to reference this dictionary, eliminating object generation altogether for objects that have previously been defined. Also, leveraging parallel processing technologies in output-stream generation is highly desired; being able to efficiently distribute the Objects Dictionary across multiple collaborating output-stream generators reduces coordination bottlenecks and is critical for exceptional performance in processing such jobs for output.

SUMMARY

The emergence in the mid 1990's of full color digital printing and the significant improvements it brought in color quality, speed, and substrates, opened the door for highly creative VDP. However, the forms-based solutions for VDP, which were the norm at that time, were creatively restrictive thus prevented using color digital printing to its full extent. Hence another type of VDP metaphor was needed. Out of this need emerged

[9] The output-stream generator uses an Objects Dictionary in order to manage the process of avoiding re-rendering of a reusable element of the document by the layout software. This dictionary is different from the Objects' Cache that is managed by the VDP RIP. Eventually, every object in the Objects Dictionary will have a ready-to-print version that is part of the RIP's Objects' Cache.

the concept of object-based page description languages for VDP (OB vPDLs), along with document models for VDP that ignited the imagination and brought highly creative VDP to reality.

RIP'ing OB vPDL is more demanding than RIP'ing forms-based vPDL, however computing technology advancements helped closing the processing-speed gap between the two. Regardless, there are professionals who prefer the "good and trusted" forms metaphor for VDP (or other, similarly restrictive, metaphors), which is essentially machine-friendly and not creative friendly like the OB vPDLs are. Such tension between human-centric and machine-centric focus is very common in technology developments. This was the case with Object-oriented programming, and even with desktop publishing. The common pattern in these cases is that the human-centric solutions are launched ahead of the technology efficiency curve, but, eventually, the technology catches up, and we get to a stage where solutions are efficient for both humans and machines. **Object-based VDP Page Description Languages are no different**; the emergence of PDF/VT as a standard is the best proof for this.

While considering the benefits of OB vPDL workflows, it is important to recognize that not all VDP applications fit well into a document-centric view, which is the focus of this paper. Some applications are based on legacy approaches that employ assembly-line kind of thinking and concepts in documents manufacturing; such concepts may still be valid as a parallel option, even when object-based VDP will be fully embraced.

Finally, the use of dynamic objects for variable data printing helped establishing a an object model that can be common for VDP and digital media. This opened the door for a holistic view of personalization, where outbound and interactive communications — print and e-media — can all be managed within a single framework.

THE XMPIE ANGLE

For XMPie, unifying the print and e-media worlds under an object model was the catalyst for the creation of its ADOR® Technology, where media-independent objects are defined as holders of dynamic content. This media-independent objects' technology enabled bringing, under the same roof, variable data printing and e-media. It became the cornerstone of XMPie's breakthrough solutions for integrated cross-media publishing.

One could argue that XMPie pushed the envelope by enabling a new era in 1:1 marketing where media is just a vehicle of communication and not a goal by itself — an era in which 1:1 marketing strategy can be defined independently of the various media channels that will be used for communication and interaction. This eventuality brought personalization truly into the mainstream as an established and recognized discipline rather than a niche, media-specific, discipline.

ACKNOWLEDGEMENTS AND PERSPECTIVE

Israel Roth and Reuven Sherwin helped the most in developing VPS, where we formalized many of the concepts that are now summarized in this document. Later they were my co-founders at XMPie, and for their professionalism and friendship I am deeply grateful. The team at Scitex, at the time when the early stages of the Creo Controller took place, helped validating many of the concepts and certainly contributed to the refinement of the VPS language and its concepts. Our Adobe partners, at the time, indirectly embraced our direction with VPS although never adopted it, explaining that VPS was based on Postscript and they were heading the PDF direction already (note that it took them more than 10 years to finally bring PDF/VT — their PDF-based OB vPDL).

Combining these ideas with the upstream workflow of desktop creation started with the development, at Scitex, of Darwin, and I am thankful for the work of that team. Developments around the ideas of objects continued tenfold when we founded XMPie. This is where the novel ideas of object-model for the print world converged with the idea of creating integrated cross media technology and solutions. XMPie is where these ideas were refined and advanced, and I am indebted to key members of the XMPie team for their innovation and drive for advancing these abstract yet unifying concepts we came up with.

During our Scitex days and first few years at XMPie, Israel, Reuven, and I were heavily involved in the PODI workgroup (PPI) that created PPML. It took us quite an effort to convince the PPI team that we need to focus on a PDL and not a complete solution from creation to production. Scitex agreed to contribute VPS to the PPI effort. The first XML formulation of an OB vPDL was done by Peter Davis from Bitstream (Pageflex), as part of the PPI workgroup efforts; we felt that it was pretty much XML encoding of VPS, to which Pageflex had a license as a partner of Scitex. The XML encoding helped align the diverse PPI membership around what later became PPML and certainly helped drive the notion of objects in this context. PPML had an important role, however, due to its vast number of dialects, which simply reflected the diverse nature of the members of the PPI working group, it never had the chance to become the encompassing standard that everyone was hoping for. In parallel, Adobe were developing further and further their PDF technology, and now with the PDF/VT they have a good chance of finally bringing to our industry the standard OB vPDL that we all need[10].

At the early years of VPS, while at Scitex, we were able to engage upstream providers, such as Atlas Software, GMC, Bitstream (Pageflex), and Document Sciences, to become emitters of VPS so that their software will be able to drive Xerox color digital printers that were driven by the Scitex (later known as Creo) controller. They certainly helped us validate and refine the OB vPDL concepts, and we certainly thank them. Also, we've learned, much later, that in quite a few advanced printing shops people programmed jobs in VPS, which is a phenomenon very similar to what happened with Postscript when it was first introduced. Their experiences, indirectly, helped establishing the validity and value of OB vPDL.

[10] During the years of the PPML evolution, Kodak's Nexpress were pushing a PDF-oriented flavor of PPML called PPML/VDX.