

This article appeared in the  
February 2004 issue of



Subscribe instantly at  
[www.bijonline.com](http://www.bijonline.com)

- Free in the U.S.
- \$18 per year in Canada and Mexico
- \$96 per year everywhere else

# XML e-Forms: Right for Your Next Project?

By Mike Rowling

New XML technologies—Web services, industry-driven XML standards, and service-oriented architectures (SOAs)—are proving cost-effective in integrating back-end systems. Where they fall short is in connecting people to these systems. Greater productivity gains are usually found in making people more efficient. When people can make better, faster decisions, an organization can be more productive, responsive, and competitive.

XML electronic forms are a natural front-end to XML-based business processes. Many developers are unaware of “e-forms” or see them as a niche technology—which, traditionally, they’ve been. Modern e-forms are optimized for rapidly automating forms-based processes. As a development tool, they’re competitive with HTML forms, which are more limited, and with compiled client-side technologies (e.g., Visual Basic, Java), which are more powerful but labor-intensive. Their nature as a front-end to other systems has meant that they’ve been among the first to incorporate integration technologies such as XML and Web services.

The e-forms market is a competitive one, with a major new entrant (Microsoft, with InfoPath 2003), a new

World Wide Web Consortium (W3C) standard (XForms), and pressure from non-XML forms vendors (Adobe’s enhancements to portable document format [PDF] forms). This is a good time to consider XML e-forms as a way to streamline the exchange of information between your systems and users.

## What’s an e-Form?

e-Forms are similar to HTML forms in that a template is defined in a design tool and interpreted and rendered in an end-user application. However, e-forms are specialized and easier to use. They typically offer:

- A toolset with native support for common forms functionality (e.g., input masks) that, in HTML forms, must be custom-coded in JavaScript
- Security features, including encryption and digital signatures
- Offline use
- A predictable onscreen and printed appearance, as the e-form filler has no cross-browser issues
- Data instances richer than the name or value pairs of HTML forms.

In some architectures, the XML e-form is a document that carries data

through a process, changing its presentation, depending on the user or the form’s state. In other architectures, the template and the data are separate entities and, as the data moves through the process, different templates are used at different points. The former treats the e-form as a document, which makes it a stronger business record useful for regulatory compliance and auditing purposes. However, this requires transmission of the entire document through the entire workflow and breaks the normal separation of presentation, logic, and data.

## e-Form Evolution

The first e-forms were print-on-demand products. An organization’s forms department would distribute electronic templates that end users would open, send to their network printer, fill in by hand, then fax or mail to the next point in the process. Although more costly per form than bulk printing, this approach eliminated the need to store and ship paper forms (many of which never get used) and often resulted in net savings.

Following print-and-fill came fill-and-print. Users could complete a form on screen before printing it, making the form easier for data entry clerks and doc-

## business integration journal takeaways

### BUSINESS

- e-Forms are specifically designed to automate forms-based processes, reducing manual effort while improving data quality.
- e-Forms, particularly XML e-forms, can leverage and extend existing infrastructure and business systems, increasing their utility.
- Some e-forms create auditable records, helping organizations automate processes and improve information management.

### TECHNICAL

- e-Forms are, for many applications, a cost-effective alternative to HTML forms, Visual Basic, and other custom development tools.
- Many (but not all) e-forms can have XML templates, hold XML data, and interact with XML Web services.
- Microsoft has put XML e-forms in the spotlight; there are now many viable products and technologies to consider.

ument scanning systems to process. It also supported embedded data validation and led to integration with other systems, such as databases, via their LAN.

The advent of the World Wide Web spurred the growth in the e-forms market. e-Forms vendors found themselves competing less with each other than with internal IT groups excited about the possibilities of HTML and server-side scripts. In response, e-form products were adapted to the new infrastructure. They worked as plug-ins within browsers and offered client-side support for HTTP and e-mail; they boasted functionality beyond HTML forms, such as digital signatures. e-Forms have continued to evolve while HTML forms have stagnated. The most obvious example of this is XML adoption. While few Web pages are written in eXtensible HyperText Markup Language (XHTML), three years after it became a standard, e-forms technologies have adopted XML as their data format and, increasingly, for the template format, too.

#### Understanding XML e-Forms

Is that e-form an XML e-form? The vendor's answer will always be yes; but you should ask if you could treat that e-form like any other XML object. That is, can custom or third-party applications read and modify it? An XML e-form is one that isn't just defined in an XML format; its schema is published and the template is accessible.

e-Forms generally support XML data. Originally, the data was exported in the vendor's format, which typically reflected the organization of the form (e.g., "<Form><Page><Field>..."). Recently, however, the industry has switched to supporting arbitrary XML data conforming to any schema. Client-side validation of form data against schemas is becoming a standard feature.

Less common are e-forms that use XML to define their templates. This typically includes presentation (e.g., the placement and properties of form fields and labels) as well as form metadata (e.g., version numbers, routing information). XML templates commonly carry non-XML content, such as JavaScript and graphics files that the template references.

The value of having a template as XML depends on how it's processed. If the e-form never leaves the vendor's product ecosystem, the format doesn't matter. But if the template may be accessed, mined, and enhanced with

other best-of-breed applications, then its XML format has real value.

Consider a national insurance company wanting to use a portal to deliver a set of 30 forms to its agents across the country. Each agency requires their logo on the forms, and each state requires specific information to be included in the form. Certain transactions may require multiple forms, which the insurance company wants to bundle together into a single compound document with a cover page listing the contents.

These objectives call out for a content management system (CMS) that can build compound documents (integrating the correct version of the required template[s] with state- and agency-specific content) then populate them with known data and serve them up in real-time. A CMS typically has general text-handling functions that can work with HTML and XML, and many are adding XML-specific functions (e.g., eXtensible Style Language [XSL] transformations). But trying to accomplish the above feat with e-form templates built in a proprietary format would likely require custom-coding and/or additional software from the e-form vendor, and may limit the choice of CMS.

#### XML e-Forms and Business Process Integration

The previous example reveals one benefit of XML e-forms: dynamic document assembly by various niche tools. XML e-forms also support process integration by allowing data and documents to move from point to point within a process. Each person or application can use or modify the form as required, then forward it on to the next point. The e-form is a message moving through the process (from the initial trigger event to the final state) that both people and systems can use.

Consider the processing of an auto accident claim. The initial claim of loss doesn't come to the carrier, but to the agency that sold the carrier's policy. The agent can look up the customer and his policy on the carrier's portal and open an XML e-form to file a claim against the policy.

The e-form can be dynamically assembled and populated with customer data, and can then guide the agent through the carrier's process for handling claims. A wizard-style series of pages can ensure the agent asks the right questions (using logic to skip anything unnecessary). Interaction with the carrier's database of auto repair shops can occur through Web services calls, ensur-

ing the agent doesn't waste time opening multiple browser sessions and that the resulting information is captured in the e-form.

When the form has been completed, the agent may sign and attempt to submit it to the carrier. If the form is incomplete (it can be validated against an XML schema file), the agent may save it to the carrier's portal for completion later. (Some XML e-forms may be saved to a local drive, too.)

The e-form has provided a standard process for customer service and data that is captured in a consistent format. Although this transaction had multiple pages, it required little interaction with the server. But like an HTML form, because it's centrally served for each transaction, deploying updates requires only posting a new version to the portal.

The carrier's application server receives the XML claim e-form and can validate it for completion against the policy and claims databases before adding a timestamp and forwarding it to the claims processing system (CPS). A nice feature of XML e-forms is their ability to carry multiple XML data instances. Process metadata—to manage and route the form and audit it afterward—can accompany claims information. Here the application server timestamp would go in the second data instance.

The CPS is essentially a rules and workflow engine sitting atop a CMS and interacting with other systems. It can check the original claim form in the CMS and declare it to the records management system (RMS) to ensure it's destroyed in accordance with the firm's retention schedule. (As the state of the claim changes, the record's classification in the RMS is also updated automatically.)

At this point, the process of initiating a claim has been completed, the data is captured in a format that people and applications can read, and the artifacts are managed and auditable. Now the CPS can analyze the claim and trigger the next process based on the claim data.

If the claim was for a car accident, additional information and specific servicing is required. A new adjustment e-form, populated with the data instance from the original claim e-form, is created and routed to a queue accessed by adjusters working in a call center. One of these can open the e-form and handle noncontentious service issues, such as determining the auto body shop the customer used and arranging payment, and perhaps capturing preliminary bodily injury claim information. The form can

**While XML e-forms are ideal tools for business process automation and integration, they carry costs beyond the sticker price.**

also have a list of actions, describing what the adjuster has done and what needs to be done, that can be submitted to form part of the claim record.

If the claim is simple enough for the adjuster to settle, they can close the claim, triggering the generation of a check and form letter. But if the injuries in this claim aren't simple to settle, automatic business rules or the adjuster may route the form to a senior adjuster, who can do a more detailed investigation.

The senior adjuster isn't tied to a call center, and can work remotely, downloading the XML e-form to a laptop. The senior adjuster can visit the accident site and the damaged car, take digital pictures of both, and attach them to the claim e-form.

Police and medical documents may arrive by mail and fax. These can be scanned in and added to the claim record. XML e-forms are used here, too. Once the document has been scanned in, the document capture clerk opens an e-form, attaches the image files and, through Web services, finds the matching claim and adds the metadata to the e-form. The clerk then submits the e-form to the CMS, which uses the metadata in the form to appropriately classify the attached images.

Once the senior adjuster has sufficient information to make his decision, he can open a claim resolution form with the claim data and supporting documents, sign it, and submit it. This final document would trigger the automated process to write a letter explaining the decision, include a check if required, and update the CPS, CMS, and RMS as needed.

#### Applying XML e-Forms to Your Enterprise

While XML e-forms are ideal tools for business process automation and integration, they carry costs beyond the sticker price:

- They are of limited value without an infrastructure of applications on the front-end. The integration points between the e-forms and those applications must typically be built from scratch, although Web services are simplifying this.
- The forms themselves must be managed and delivered, which may require a new or expanded portal or CMS.
- XML e-forms are likely to be a novel technology to your development team, adding to the risk of project failure.

These issues aren't unique to e-forms, but because e-forms appear to be "simple" technology, the effort required to understand and use them is often underestimated.

Typical decisions required in XML e-forms implementations include:

- How much logic should be embedded in the forms and how much should be in the server?
- Should a form's schema be form-centric (i.e., based on the form's fields, which are typically already accepted within the organization) or data-centric (i.e., be a subset of the enterprise data model, which is often incomplete and requires multiple stakeholders to agree)?
- Should industry-based standards be used and, if so, do they need to be

extended to meet the enterprise's requirements?

- How securely should forms be handled?
- Does the form have value as a document, requiring that it be kept intact for subsequent reviews, or can it be destroyed as soon as the data is extracted?
- If the form is a document, what is its life cycle?
- Will users have to work offline?
- Do the e-forms have to look exactly like their paper equivalents? If so, do they have to behave like paper, or can they change size and shape to accommodate variable user input? Will they ever be printed, and what quality of printing is required?
- Are the forms strictly internal, or will they be used by suppliers and customers?
- Will e-form routing be strictly determined by business rules, or should users be able to decide who needs the e-form next and route it to that person?

To answer these questions, it's necessary to not only understand the requirements of a project but also to understand how various XML e-form technologies address these issues. The user community for all XML e-form technologies is relatively small, so e-form-centric expertise is best acquired from the vendors, which typically offer presales consulting.

#### Conclusion

XML e-forms are built for streamlining the interactions between people and systems, and are worth investigating because of their lower development costs and operational benefits. They provide data about the process, allowing both real-time reporting and historical analysis of costs and process bottlenecks. Finally, they provide a strategic foundation for further enhancements to an enterprise's processes.

Given the proven benefits of both existing and emerging e-form technologies, this is a good time to consider it for the exchange between your systems and users. **bij**

#### About the Author

Mike Rowling is technical product manager at PureEdge Solutions, where he's responsible for the architecture of the Integrated Development Kit for IBM DB2 Content Manager and the XForms design product. He previously worked for a European vendor of ERP solutions.

e-Mail: [Mike.Rowling@PureEdge.com](mailto:Mike.Rowling@PureEdge.com)

Website: [www.PureEdge.com](http://www.PureEdge.com)