FORMAT FOR ELECTRONIC FORMS

In the past, electronic forms processing has been handled on an ad-hoc basis: each paper form was Web enabled by creating custom software. However, the process of editing an electronic representation of a paper form, verifying the data, and viewing the results can be generalized. By producing a universal set of tools for editing and viewing electronic forms, and specification files that capture the essence of each form, the process can be simplified by orders of magnitude. The forms processing tool set can be created and maintained once for use with thousands of easy to produce specification files.

For over one hundred years, paper forms were the only method available for processing the flow of information between Nebraskans and their state government. Over the last five years, information processing with the public has been revolutionized. By employing the World Wide Web as a technology for enabling E-government, we have developed the skills and tools to effectively create electronic access to government services. Many functions that could only be accessed through a paper document filled in with typewriter or pen are now performed with a computer. The savings in time and dollars to taxpayers and citizens is beyond dispute.

However, up to this time, government services have generally been automated one at a time. Each paper form replaced (or augmented) by an electronic counterpart has required:

* An HTML representation of the form (or forms) involved

* Software to process the submitted forms

* A storage and retrieval mechanism for the resulting information

* Error checking code to make sure HTML fields are filled in properly

This collection of markup and code, built up in place of each automated form, must be created and maintained by trained, experienced programmers. The code must be modified to accommodate even a minute change.

While this approach works well for a small collection of processes, it is unlikely that this method will scale for the thousands of forms the State of Nebraska uses. The time required for the creation and maintenance of a code base for each form certainly makes this practice unattractive, and may make it untenable.

Furthermore, the automated process is tied to the technology put to use at the time, making modernization difficult. For example, a programmer working only a few years ago may have made a wise choice, based on the mature technology of the time, by employing Perl, Informix, and HTML 2.0 to automate a previously paper-only process. These technologies are still viable today and are likely to be useful far into the future. However, today we might wish to make use of Java, Oracle, and HTML 4.0. Under our current development practices, the only way to do this would be to scrap the existing system and rewrite it with today's technology in mind.

Clearly, a more generalized approach is required if we are to quickly create and efficiently maintain a large base of electronic forms. So, rather than applying individual technologies and skills to individual forms, it makes sense to create an architecture for electronic forms automation.

Under the electronic forms architecture proposed by NOL, the current practice of creating HTML documents and a code base for each electronic form will no longer be needed. Instead, each electronic form will rely on a small collection of specification and data files, and all electronic forms will be processed by a suite of generalized tools and applications. However, the specification and data files will also be easy to use with custom software written in any language, should their be a need to do so.

The specification and data files required by each electronic form will include:

* A master "blank form" specification file. This electronic document will contain an entry for each blank on the paper version of the form, type information for the blanks, the text used to prompt the user for the data for each blank, and help information for the user.

* A formatting specification file, used for converting the completed form into a data structure requested by the agency for convenient incorporation into the agency's information systems.

* A constraints file. Any limits on the data to be entered into the electronic version of the file will be placed here. For example, a constraint could be specified to limit a social security number blank to exactly nine digits.

* One or more data files containing an image of the original paper form, and one or more specification documents containing placement information. One placement file will be needed for each image file. This placement information, combined with a data file for a completed form and the corresponding image file, will be all that is needed to produce a graphical representation of the completed form. The resulting image will bear an exact resemblance to a completed, typewritten paper form.

The suite of electronic forms software will include (but need not be limited to):

* Editing software, allowing users of the form (both among the public and within state government) to fill in the form blanks. A Web interface will probably be the best choice when designing this editing software. The editing software may also consult the constraints file to assure that no incorrectly formatted data is introduced.

* Constraints checking software. The completed form can be checked against the constraints file to make sure no errors were introduced during the data entry.

* Image generation software. This software will combine the completed form data, the image of the paper form, and the layout document to produce an image of a completed form, just as it would look if the process were done on paper only.

* Formatting software. The formatting specification will be used as a guide to generating output in a form appropriate for the agency's use. For example, the data filled in to the form's electronic "blanks" could be output as comma separated values for importing into a RDBMS. More than one formatting file could be provided if needed.

Nothing about the above software restricts it to one programming language or platform. Once the necessary specification files are in place, any version of the above tools can be used to produce and process the data files. The software could be reimplemented in the future to take advantage of the latest technological advances -- the spec files will be just as useful with Java as they are with Perl or C. Also, implementations of the electronic forms processing tools can be used concurrently. The public might process the form using an HTML interface, while the same form could be further edited by state agency staffers with a custom-built, Java based tool.

As needs change, tools and features can be added to the suite of forms processing software. Suppose that one agency needs to be able to process their forms from within Microsoft Excel. Software could be written to turn a standard data file into an Excel spreadsheet. Because the form data is in a standard format, the software could be used with any properly formatted data file. A set of macros developed for the unique needs of, say, Banking and Finance could be used by the Department of Revenue or the Nebraska Library Commission. The standardized format would allow this new software to work with forms the designers of the Excel macros had never seen or heard of.

Forms, be they paper or electronic, have always been a part of state government and are with us to stay. When carbon paper was replaced with photocopy machines, the forms they duplicated stayed essentially the same. By solving the general problem of electronically enabling paper forms, we can continue to take advantage of the latest technology, no matter what the future might bring. Let's create electronic forms that can take advantage of new technology just as readily as their paper predecessors.