Why HTML is a Strategic Dead End for Business Transactions and E-Commerce

Wayne V Duquaine

Grandview Systems

HTPS 1999

3270 is dead. The only question is how long will it take to bury the body – 10 years, 15 years, more ? HTML, the new hot display technology, is immensely popular today. However, like its 3270 cousin, HTML is ultimately as strategically dead as 3270 is. HTML suffers from the same ultimate fatal weaknesses that doomed 3270 – it is principally a display technology, that treats intelligent devices like PCs and PDAs as dumbed down display devices.

While HTML can be extended to do "dynamic" things like fancy drop-down lists, user input validation, animation of client screens objects, etc, this requires the use of outboard programming languages such as JavaScript or VB-Script running on the client. While this is a welcome relief from truly awful approaches such as HLLAPI, the net effect is that all this huffing and puffing and fancy programming still revolves around a dumb display model. The wave of the future, particularly in business-to-business, database to database, messaging system to messaging system, and other advanced forms of these such as E-commerce, is not well served by trying to promulgate data around the network using HTML (aka display) formats. Trying to make such systems inter-operate using HTML as a basis of interoperability will ultimately result in "screen scraping phase II", only this time using HTML tags rather than 3270 attribute bytes. HTML provides a pretty face, but a lousy system-to-system transaction environment.

CICS has recently added HTML support to its bag of tricks. You can re-compile your 3270 BMS maps and have them generate HTML formats rather than 3270 formats. While this is technically interesting, it is a best a pyrrhic victory. It only underscores that CICS continues to be cursed by its 3270 legacy.

The world is moving increasingly toward intelligent applications consisting of: software components that communicate with each other, "active databases" that communicate with each other, objects that communicate with each other, messaging systems that communicate with each other, and so forth. In such a world, display based technologies such as HTML are increasingly dysfunctional. This is because:

1. Intelligent system-to-system interaction requires information about the data structure of the information sent. Using old-style "application embedded knowledge" of the data (e.g. COBOL copy books of what the data "should" look like), breaks down as hundreds of new component based and object based applications come online. It becomes an N x N type of problem to make such old application technologies work together. Not every bank or shipping company in the world is going to be able to publish its display formats and program for everyone else’s formats. The complexity becomes overwhelming.
2. Automated translation of one data format into a different data format is difficult, when only display format information is present.
3. Trying to make screen-scraping techniques work with new applications has proven to be a mess.
4. Maintenance becomes a major problem. Even minor changes in display format often causes existing screen scraper or logic that depends upon fixed format/fixed position fields to fail.

For E-commerce or other business-to-business data exchange to work, some form of any-to-any connectivity is a required fundamental building block. To make any-to-any connectivity work between companies, some form of self-description of the data records/data objects/data messages being sent from one organization to another becomes imperative. Simple, "I can get TCP/IP to work between two companies" level of wire connectivity is insufficient. Application level connectivity is required to make automated business-to-business transactions feasible. For any-to-any, plug-and-play application level connectivity to work, the data must be self-describing.

The wave of the future will be application designs that can exchange self-describing data and forms. There are several potential candidates, such as DRDA, IIOP, DCOM (on a good day), XML, and so forth. The most ubiquitous and widely available cross-platform solution is XML (Extended Markup Language). It provides a completely self-describing format of the data or message being sent or received. XML meta-data formats (DTDs) for a message can be either included in-line or centralized in a repository and referenced via a standard web URL.

E-commerce and Web applications using XML are just beginning to appear. A standardized programming interface to process XML documents/forms/records, called DOM (Document Object Model) has recently been approved. Microsoft’s Internet Explorer 5 (IE5) browser comes with XML and DOM support built in. Microsoft’s Web Server-based ASP technology can be used to easily generate XML output, and process incoming XML input. "Open Source" based solutions exist on the internet to process XML, and Apache Web Server based solutions such as JSP (Java Server Pages) can generate and process XML just as easily as Microsoft’s ASP.

Truly powerful applications can be built using combinations of JavaScript and XML. Not only can the data and its format (XML) be shipped to another system, but the associated processing logic to validate data entered into the record or form can be shipped along as JavaScript as well. XML can be used to mediate between Database row-based output, asynchronous message records, and other types of business forms and data. And because XML is self describing, applications are insulated from changes in the underlying record format.

Business-based TXP systems of the future will need to incorporate XML as part of their backbone services technology. The ability to read and understand an XML description of an incoming record or object, and the ability to easily generate XML to describe an outbound record or object, will be a major requirement for system-to-system, business-to-business transaction processing. Intelligent browsers will have XML built in, allowing XML to be used bi-directionally all the way up and down the business hierarchy: from PDAs to PCs to Web/App Servers to Transaction processors. Final output to human end-users will be achieved via XML to XSL or XML to CSS rendering engines, operating on the local PC or PDA. All system-to-system interactions will use the self-describing forms embodied in XML. As XML-enabled browsers (such as IE 5) proliferate, HTML will begin its long, slow descent into emulating the 3270’s legacy – another dead-end display only technology.