

A popular development method, currently, is to develop, so called “web applications”. These web applications suffer from fundamental design problems because of the technologies they are based on. The very concept of a web application is flawed and should be avoided.

A web application is a piece of software, that runs on a HTTP (web) server and is written in PHP, ASP and to a lesser degree, PERL and transmits an HTML page to a user via a web browser. What separates a web application from a simple CGI program is that a web application is usually integrated with a database (especially a SQL-based database) and a web application performs many functions while a CGI usually has a simple data collection or extraction function. Popular web applications include on-line banking, *webmail*, forums and various company-specific applications deployed on corporate Intranets.

The first problem the infrastructure on which the applications are based. The HTTP protocol was designed for the request of static documents from servers over the Internet. The original method of requesting documents based on the GET method which was basically an accepted convention that any data after the question mark in a file name is not part of the file name, but command line parameters to that program. The protocol was later expanded to include a POST method to transmit more complicated data. Although, this provides a reasonable method by which to communicate with a server, it is awkward for large quantities of binary data and slower than native applications. If it were simply these infrastructure problems, there would be not a severe impairment. The real problem is that HTTP is designed to be stateless. Each request is completely independent of any previous or subsequent request from the server. Any kind of connections between these requests must be assumed by the server or client though some kind of token exchange. Since the system was not designed for this, there are many gaps in this kind of programming logic. These sessions are error prone and, in some, there is no way to forcibly expire a session and can expire erroneously. The underlying infrastructure was never designed to support web applications in anyway and the additions are less than satisfactory.

Because of the way the web evolved, a programmer cannot guarantee how the user’s web browser will render the resulting page. This is a more general problem with the web. Although the World Wide Web Consortium (W3C) has created a series of standards to ensure that pages will render similarly under different browsers on different platforms; unfortunately, no browser actually complies with the W3C’s specifications. Although even browsers have faults, Microsoft’s *Internet Explorer* is the least compliant, yet is the most popular browser leaving programmers with the dilemma either to write non-compliant pages that will render correctly on Internet Explorer with its some 95% of the market share or to write W3C compliant pages that will not work correctly under Internet Explorer. Most programmers choose to either work only with Internet Explorer, write a page that is almost compliant so that it will mostly work under most browsers or try to detect the browser and maintain several copies of the page for different browsers. This is absolute foolishness. A C programmer would be out-raged and refuse to program if the programming libraries to output data was different on different systems. Each browser also has unique interpretation of the JavaScript language that can be embedded in HTML pages. Again, the W3C has created a standardised “ECMA script”, but again, it has been promptly ignored.

It is a pity that JavaScript is so paralysed because it is the only part of a web application that can dynamically interact with the user. Because the standard was designed to fetch static pages from a remote system, the page cannot interact bidirectionally with the server, it can, at best, request another page. This leaves the user with awkwardly designed pages that either constantly need to fetch new copies from the server or large pages that must download all relevant data and have JavaScript do much of the processing. In either case this leads to programs which are non-intuitive and have illogical flows relative to programs designed using standard widget toolkits.

In short, the foundation on which web applications are built was never designed to support them which results in programs of poor quality. I realise that web applications cannot be removed completely since some functions, like on-line banking and webmail, have no alternative means to be implemented. However, companies should do everything possible to avoid creating and using web applications.