Some developers strongly feel that these new technologies are coming along much faster than they can be absorbed. There have been many heated debates on the extra aggressiveness with which Microsoft is releasing new products. In many of the offline discussions we have had upto now, most people feel that developers are not getting enough time to absorb existing technologies and "keep pace with MS". People are still struggling with master pages and partial classes, and we have AJAX, SilverLight, WPF, etc.!

Many developers feel that there is simply too much to grasp in too little time, considering the fact that many clients are still using VS2003 and are refusing to upgrade due to reasons such as lack of funds, apprehension of going with new and untested technologies such as WPF and Silverlight, lack of experienced programmers, and so on. Some customers are also confused and are not sure how these new technologies can benefit their business!

We should understand that none of the new technologies were created to be "silver bullets". They have been added to give developers options to chose from, to reduce development time, and to be more effective. These technologies should be used in the right architectural context instead of blindly following them, which can lead to a greater risk through poor implementation. All changes are good, but we need to understand why the change is needed and how it will help us in balancing the advantages and disadvantages.

We have thousands of books, online articles and tutorials on how to use AJAX, LINQ, WWF, and WPF in ASP.NET, but there are still very few online articles and limited books that focus on what architecture to use, and in which ASP.NET application. Because each project is unique in its own way, we can never use a copy-paste solution. The important thing to bar in mind when learning application architecture and design is that there are no strict rules, only guidelines. And these guidelines were developed based on the experience gained over years of work by developers on different projects.

Upcoming latest technologies should not be mistaken as the means to develop better applications. Lets go back to the pre-ASP.NET years for a moment. In those days, classic ASP was very famous. There were many big, famous, and stable applications in classic ASP 3.0. It was difficult to create an object-oriented application with classic ASP (compared to the intuitive way, in which we can do it so easily now in ASP.NET), but good programmers used classes in ASP as well, adopting elements of object-oriented re-usable design. A better platform, such as ASP.NET, did help in building websites that could support a better architecture, but the power to use it in an efficient way still lies in the hands of an experienced programmer.

Just as ASP.NET was a major stepping stone in web development, AJAX enhanced the UI experience along the same lines, providing a user-friendly experience while browsing websites, and LINQ was introduced to revolutionize data access. But still there are numerous robust and popular websites in ASP.NET not using any of the new technologies. This means that the key to building a good website can never only be learning and absorbing the latest technology out there, but also how you put it to use—how you make these technologies work for your project in a comprehensive way.

If one knows how to write clean and maintainable code and use efficient programming techniques to create a good stable architectural platform and application structure, then technology will supplement the design. Without a stable architecture and good coding practices, a programmer might use the technologies in a haphazard manner, creating messy code and junk websites. But once we understand basics of the application architecture and the different design patterns, then these technology tools become our assets.

Technology and Art

Unlike coding, which demands strong logical skills, application architecture and design is more of an art, and it takes time and experience to become a good architect. For example, it takes a very good and experienced designer to create a unique and attractive design for a car. Once it's done, the assembly line can create millions of units of that model using the appropriate machines and tools for the job. Similarly, it is relatively easier to understand and code in ASP.NET, but it can take some time for even an intermediate developer to be able to understand and design the pros and cons of the different architectural options that might suit a given web application. And unlike coding, there are no strict rules in architecture. A design which might not work for some projects can work perfectly well for others. That's why it might take years of experience to develop an eye for good architecture and design. This, coupled with the fact that each application is unique in its own sense and warrants its own design and architecture, can be confusing for developers when deciding what is best for their project.

Therefore, architecture is one thing which requires patient understanding, as well as creativity in order to be able to adapt and innovate according to a project's needs.