

## Re: Architectural feedback

**Source:** <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2005-01/1049.html>

---

**From:** William Stacey [MVP] ([staceywREMOVE\\_at\\_mvps.org](mailto:staceywREMOVE_at_mvps.org))

**Date:** 01/10/05

Date: Mon, 10 Jan 2005 13:44:28 -0500

wow. First, I would look \*seriously at some existing accounting packages and/or CRM that your company could buy, that may be extensible or have existing apis/web service points to extent for your specific needs. Writing a complete accounting system is just a massive task and many have failed so why recreate the wheel? I would keep looking first.

However, say you need to do xyz and need custom inhouse function. Below is a big generalization and naturally the options are long and things are complicated. Lets forget the DB for now and just say that is an implementation issue and can be abstracted with your SOA. Let also assume you have one server and one client. I would use WSE using tcp or http via IIS and create my Business layer/SOA contracts. CreateCustomer(), UpdateCustomer(), etc, etc. Start small and get the base functions you need. Create your smart clients as std winforms apps using c# or VB. They will call web services server to get lists, updates, etc. As the business layer is web services, you can use console clients, windows clients, or web clients and they all call the same business logic/web services. Just your presentation layer changes. I said forget the db, but SQL 2005 could change this a bit. SQL 2005 can now host xml web services and CLR .net assemblies. So in some respects, it can now be a one-stop-shop to host your business tier and have ready access to your data as your inside the server. Naturally you can leverage all the transaction ability of the server within your business code and leverage things like the new Broker which can give some cool options and reliability in your code. You can also start coding with it now for proof of concept stuff and prototyping. I may be well worth the effort to look hard at it now and see how much logic you want to keep in the "server" and what you may want on other web servers that just call sql DB in normal manner (i.e. ADO.Net, etc.) With integrated web services, direct inproc DB access, and CLR hosting, it makes a powerful thing to look hard at now.

Again this is a big topic and you really want to get some good books on three tier arch, etc. Good luck!

--

William Stacey, MVP

<http://mvp.support.microsoft.com>

"John Spiegel" <[jspiegel@YETANOTHERSPAMHATERc-comld.com](mailto:jspiegel@YETANOTHERSPAMHATERc-comld.com)> wrote in message news:#wQDPuz9EHA.3980@TK2MSFTNGP11.phx.gbl...

## microsoft.public.dotnet.general: Re: Architectural feedback

> Hi all,  
>  
> I'm in the early stages of designing a system that we'll use in-house and  
am  
> looking for opinions and suggestions for further reading on which  
directions  
> to take.  
>  
> Background: Though we're a small company, I'd like to design this with  
the  
> mindset of a larger company from the start. This will be an ongoing  
project  
> to encompass a lot of needs, which will generally be variations on  
standard  
> themes: Customer account management, billing, reporting and other fairly  
> common business functions. Most functionality will be accessed from our  
LAN  
> with WinForms-based apps (assume all client machines are Windows running  
at  
> least 2000). There will be some limited Web access for employees, e.g.,  
> order forms for sales staff. While we may eventually want to build some  
Web  
> services for sharing information, the general public website will remain  
> fairly autonomous.  
>  
> The nature of the company and application is that there will be fairly  
> frequent releasing of the system, or components thereof.  
>  
> Current leanings: Most of the functionality will probably APPEAR to be  
> (from user standpoint) unified under a single app. There are a number of  
> underlying components into which this apparent monolith will actually be  
> broken. Aside from various departmental boundaries that will make for  
nice  
> conceptual breaks. Utilities, e.g., common dialogs, data access and  
account  
> calculations, will be broken into various solutions. I'm thinking that  
> there will be a skeleton client app that provides the system navigation,  
> security and user preferences then uses the appropriate other resources to  
> provide actual functionality.  
>  
> A lot of my uncertainties are in determining what forms and in what  
> locations the pieces should be. For example, I'm not clear on whether the  
> core app must reside on the client or may be run from the server. Is it  
> better to leave the individual dll's on the server, or install in each  
> client's GAC? My guess is that deploying all components to individual  
> clients will give notably better performance. On the downside, that would  
> imply updating a number of machines every time we add a new or function.or  
> want to release bug fixes.  
>  
> I've read a fair amount on remoting, serviced components, and related  
> functional divisions (a 70-320 prep book, mainly), but have seen little  
that  
> really ties together when the best situations to use each are.  
>  
> TIA,  
>  
> John  
>  
>