Context: I prepared and distributed this simple guidance to be posted on the cubicle walls in a 40+ person highly-specialized development organization.

**XXXX Legacy Rewrite Application Guidance**

**By Don Krapohl, AugmentedIntel**

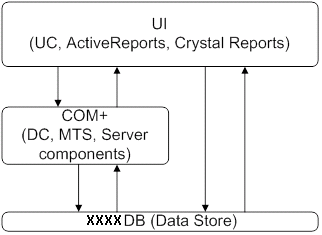
1. Purpose
   1. This document is meant to answer some basic questions that will arise as you begin development. It is in no way meant to convey the detailed logical or physical architecture, project tasking and responsibilities, business requirements, or system operational environment.
   2. This document may be updated as necessary to provide high-level guidance on the XXXX rewrite.
2. Development Environment
   1. Workstation configuration

* Vista or Server OS, attached to xxxx domain
* Visual Studio 2010 with C# and Reporting Services (SSRS),no c++, no
* AJAX Control toolkit
* Enterprise Libraries 4.
* VB6 SP5
* Crystal Reports for VB6
* For BI developers: SSAS, SSIS, SSRS
* Optional : Reflector
  1. Source Control
  + Team Foundation Server (TFS) URL is xxxxxxxxxxxxxxxxxx
  + VB6 code is in VSS at \\server\vssXXXX
  + XXXXX and XXXX will be responsible for setting up the structure and policy of source control and will enforce it.
  + At the start of a development cycle the PM will request TFS project permissions according to your assigned project/application
  + Only move source code from person to person by way of TFS
  + When ready to send working source to another person (from BusDev to UI for example) assign it to the person’s work queue.

1. Project Info
   1. Goals, generalized:

|  |  |  |
| --- | --- | --- |
| User Goals | Business Goals | Infra/Ops Goals |
| * New system contains the same functionality as it does now * Performs reliably * Is responsive (qualitatively) * No user re-training required | * Eliminate VB6 by eliminating VB6 code * Delivery by July 2011 * Maintain current system availability and reliability * Provide ability to migrate DB to Oracle * Build system to change not to last. | * Minimize network traffic * Increase security in communication, storage, and interface * Simplify deployment |

1. Architecture
   1. Legacy Architecture



* 1. Future Architecture



1. Technologies
   1. CSS for UI consistency
   2. ASPX or HTML for web pages, as appropriate
   3. C# code behind in a file separate from the aspx.
   4. AJAX to reduce network footprint and increase responsiveness
   5. JQuery 1.4.2 to simplify client-side coding and add behaviors
   6. JQueryUI 1.8.1 for UI manipulation
   7. Enterprise Library 4.1
      1. Caching
      2. Data
      3. Validation
      4. ExceptionHandling
   8. SQL Server Reporting Services 2008 (SSRS) for reports
   9. Always use the absolute most basic controls when making a design choice. Assume HTML controls and move up to more feature-rich if required (ASP.NET then AJAX, etc.)
2. Code Acceptance Criteria
   1. Code may be reviewed by one of the senior developers for acceptance. If rejected you can request review by the senior dev group as identified by Sal.
   2. Code containing variables, members, classes, or other entities named for Star Wars characters will be rejected
   3. Use common sense in naming. Use names that identify what the object or method does or contains.
   4. Only use namespaces patterns as indicated in this guide
   5. Code must compile
   6. In general, one class per file, file must be named the same as the class
3. UI
   1. UI candidates created by the developer should contain only minimal event code, if any
   2. UI developer will work with the business developer and BA to create user interfaces. UI dev is the final authority on UI to maintain consistency across applications.
   3. In ASPX pages minimize ViewState by using HTML, AJAX controls, and/or turning ViewState off as appropriate.
   4. Avoid use of inline style

Guidelines for the UI designer:

* 1. Use the standard CSS file. If you need a special style more than twice consider adding it to the CSS.
  2. Use the supplied master page
  3. Use validators consistently on input controls. Add formatting hints as appropriate.
  4. The system may be delivered through SharePoint and/or our own framework. Design for 650px at 800x600 or 850px at 1024x768.

1. Database
   1. All database changes are out of scope with the following caveat: If you believe a change is necessary it must be impossible for the application by any design or algorithm to run without the change as determined by the lead DBA.
   2. For the Data Services Developer:

* Only execute against the team database in XXXX—no local database, data, or schema should exist due to security constraints.
* Access the database only through supplied stored procedures.
* The lead DBA will be responsible for any modifications required to the database.
* Database changes are not in scope for this rewrite. All our efforts should be expended on reusing the existing structures and objects.
* For scalability make services stateless, cache domain-specific data (list of weapons, rank/grade, MOSs, etc.) using server-side caching and perform as much processing in the service layer as possible.

1. Code
   1. Make sure your methods, types and type members, especially public, have the appropriate

/// <summary> and supporting sections

* 1. Use the Microsoft.Practices.EnterpriseLibrary.Logging component for error and other logging.

1. Exception Management
   1. Do not throw an exception for the purpose of recording errors or flow control
   2. Define unique exceptions to be thrown
   3. Only catch what you can handle
2. Business rule sources
   1. Read the VB6 code to derive rules. Some rules may be implied in VB6 control behaviors.
   2. All web pages in XXXX have explanations of behaviors, rules, and law immediately following the control.
   3. The BA for the function is responsible for clearly elucidating business rules.
3. Namespaces
   1. Namespaces should have the form of system.application.module.functionality where

* System is the name of the system that namespace is designed for (XXXX)
* Application is the name of the application the namespace is designed for (RM,PER)
* Module is the name of the application module the namespace is designed for, if there is one. (PerOrders)
* Functionality is the specific technology area that is being supported for the module (Entities, UI)
* Solution Name = Library Name = Namespace Name less functionality (ie, XXXX.BusinessArea.Module.sln)