**Modernization of Legacy Applications**Untangling a complicated compilation of UI, BLL and DAL code into isolated parts.

Written by Carl Boudreau, Sr. Principal Engineer | Software Architect for Modern Clinical Delivery  
carl\_boudreau@optum.com

*If you haven’t read my article “****Start with a Wireframe! Agile is not an excuse to start coding without a plan****” I would recommend you read it and save you and your team a lot of frustrations while developing your application.*

One of the most overlooked and failed responsibilities when delivering software today is a failure to create a plan to maintain and support an application 3 years out. In my 18 years of developing solutions I have never seen a legacy application not go thru some change or upgrade after the initial delivery. Yet somehow it fell into the legacy application bucket. How and why this happens is covered in another article. This article will focus on what can we do to modernize an existing legacy application.

Most applications can be sub divided into 3 parts; the User Interface (UI) layer, the Business Logic Layer (BLL) and the Data Access Layer (DAL). Applications that fall into the Legacy category include the UI, BLL and DAL code compiled into one file. Today we learned that wasn’t the best practice, it prevented us from changing the data storage method or technologies, and that kept us locked into expensive licensing agreements.

Modernization need to be approached in small steps. Like open heart surgery we still need to keep the patient alive by keeping the blood flowing. To enable quick and easy replacement of the heart in the future we need to install quick disconnect fittings. Preparing a legacy application for modernization you will position yourself strategically. Installing quick disconnect fitting so when it comes time to replace old functionality with newer functionality it is as simple as quickly disconnecting one part with another.  


In the image above is a quick disconnect used in fluid transmission. In software we call this a “Data Contracts” and “Method”. Data Contract describes what data is being passed back and forth and the Method describes how it is being passed back and forth.

The first step to modernization is to find the integration points between presentation (UI), Logic (BLL) and the data access (DAL). Because the BLL is generally in the middle, it is best to start with the BLL and the DAL. Mark integration points where ever the BLL, CRUDS (Create, Read, Updates or Delete) data.

The second step is to choose the correct Method to transmit data between the BLL and the DAL that is simple, reliable and not going away in 3 years. For example a JSON Web Service seems pretty solid at the time of this writing. However Kafka Streaming is becoming popular to, but it is too early to tell how long it will be around.

The third step is to implement the interface between the two existing layers (BLL & DAL) and test. Remember, we are not introducing new technologies; we are just creating a quick disconnection feature here. We should never plan shift to new technologies in a legacy application in steps. Implementing a new database slowly will strain your team during the modularization effort. Your maintenance will double and the risk of failure doubles. Installing the “Quick Disconnects” features first will give your application time to flush out any issues and your team can transitions slowly and safety while keeping your application up 999’s

Once you have completed the ability to separate the layers you now can preplace older technology with new technology and never having to manage another legacy application.  
