Case Study of a VFP to Xbase++ Conversion

This session is a discussion of how Xbase++ and an SGL language extension simplified the conversion of a Visual FoxPro application to Xbase++.

This is a case study of how an actual conversion, completed this year, was accomplished in a very short period of time.

The company who hired me to help with the conversion had been in business for many years and chose one of their many VFP applications as a not only a feasiblity test but also has deployed the new Xbase++ application to theircustomers.

Attendees will receive a free copy of the SGL system that was used for the conversion.

* The appropriate steps in performing a conversion.

* What type of VFP applications can be easily converted using this techique.

* What VFP developers need to know about Xbase++ when converting.

* What VFP developers need to know about SGL programming.

* A look at the source code of the finished application.

* A comparison of the finished VFP application to the finished Xbase++ application.

* Commentary from the VFP developer about their experiences during the process.

Background

In 2009, I was invited by Steffen Pirsig to join him at the SWFox developers conference in Phoenix, Arizona. After arriving there he informed me that Microsoft had officially announced that they were dropping support for VFP and that he saw a new opportunity to embrace the VFP community with the ultimate goal of giving them a platform to migrate their applications.

So every year after that first year I attended the SWFox/SWXbase conference and helped toward that objective. Each year there was an increasing interest from the VFP developers in Xbase++ technology yet it was becoming more and more obvious that neither the VFP developers nor the SWFox leadership ready to take a serious interest in Xbase++ and actually start working on a migration project. I had my own reasons why I believed that we working toward a futile endeavor and so, in 2015, I decided I would no longer attend these conferences. It became obvious to me that VFP programmers had been so assimilated by Microsoft that they would never use a development language and environment that did not look like Visual Foxpro. Alaska Software kept making promises that they were working on all the necessary tools and instead of talking about Xbase++ 2.0 (which was still not finished) they only talked about Xbase++ 3.0 and the far future. From my own experience in working Alaska Software for over 15 years, and also working with Microsoft developers, I decided it was not likely that Alaska could ever satisfy their requirements.

Steffen Pirsig called me and asked me to reconsider, and so I did. But being a pragmatic person who doesn't believe in the insanity of doing the same thing and expecting different results, I agreed that I would try to engage the VFP community one more time but on my own terms. And so, in 2015 I chose to promote a different idea that could possibly convince some VFP developers that I could help them with their migration by utilizing a programming concept called SGL (Structured GUI Language). I introduced SGL in both Arizona and in Frankfurt in 2015 to VFP developers and tried to make the case that they did not need to wait for Xbase++ 3.0. Soon after the 2015 conference I received an inquiry from Stewart Ganser. He had attended my session and was ready to work with me on a migration project for his product - **MyFireRules**.

With the introduction of SGL, I had to convince VFP developers that they could develop GUI screens as fast or even faster than a VFP screen designer by using a declarative language. In this session, I will explain the process I went through to achieve that objective and the costs and time involved.

My Fire Rules

Here is a statement from Stewart Ganser, CPA , partner in The Rules Guys (LLC):

The Rules Guys LLC ("TRG") is a small software company that has produced a number of apps to enhance Fire & EMS software solutions from ZOLL Data Management ("ZOLL"), a subsidiary of ZOLL Medical Corporation. These apps add data quality, data analysis, data visualization & automated billing enhancements to the ZOLL solutions. These apps are written in Visual FoxPro ("VFP") and Transact-SQL.

TRG has been searching for a VFP replacement for many years and as any programmer knows this is a daunting task to say the least! They have chosen Alaska Software's Xbase++ & Donnay Software's eXpress++ as the tools and Roger Donnay as the consultant/programmer. Roger is reprogramming screen by screen with integration & maintenance turned over to TRG . This is planned as a multi-year conversion project of which 2016 is year one.

The first app chosen for conversion is called "My Validator". This app significantly improves data quality by providing real-time validation results of locally defined business rules at the time of data entry. The first phase of this conversion involved replacing the validation result screens and is shipping now. The second & final phase of this conversion involves complete replacement of the app, is currently being tested by TRG and is expected to ship before the end of the year.

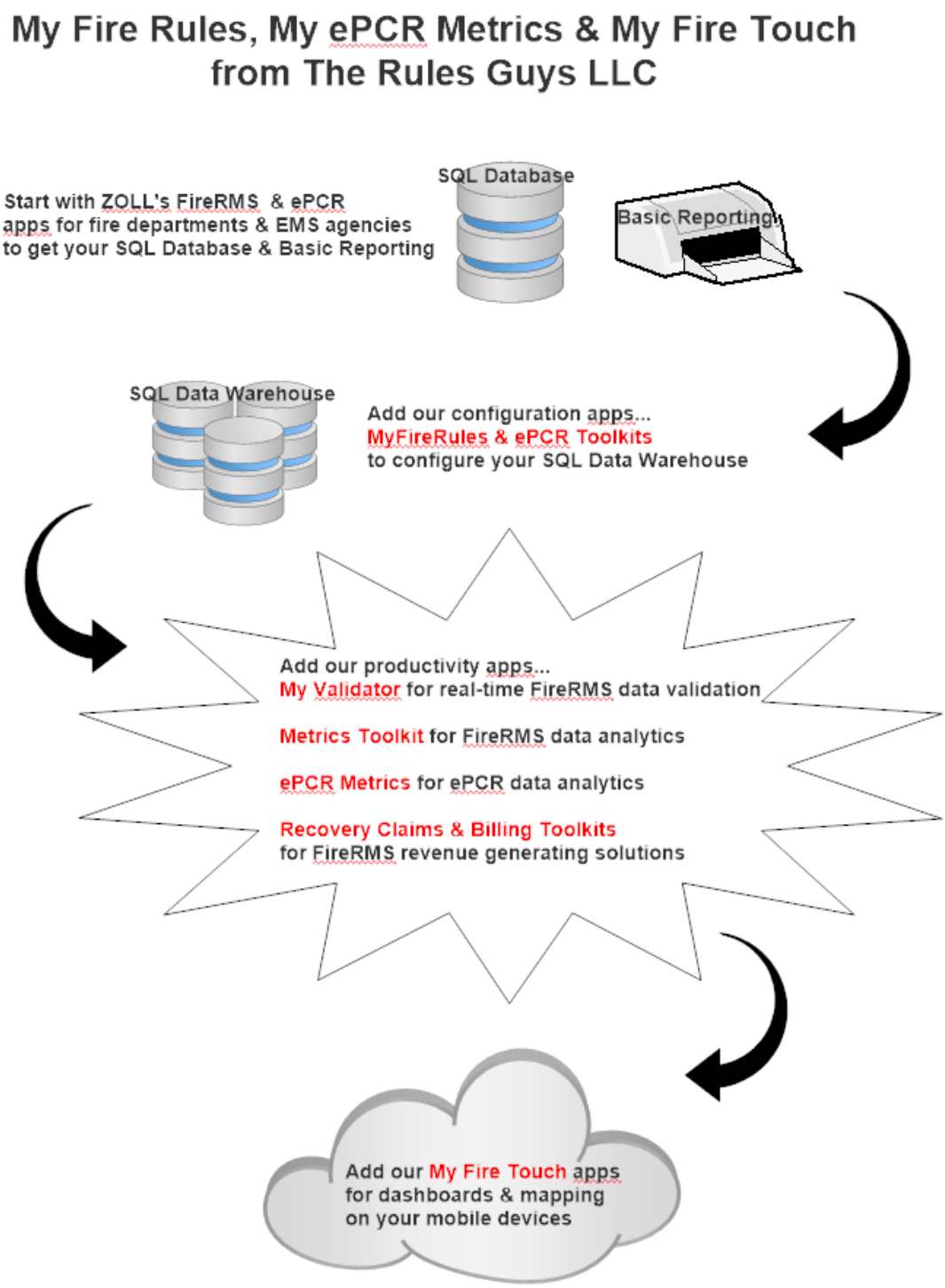
Here is a document created by Mat Jackmond, the Chief Rule Maker for My Fire Rules:

My Fire Rules was developed as an "Add-On Product for ZOLL Data System's FireRMS Product" by firefighters who, like you, have struggled with the problem of obtaining, validating and reporting upon user entered Fire & EMS Incident DATA. John McIntire, former Chief Information Officer of Los Angeles County Fire Department and respected NFA "NFIRS Data-Mining" Instructor has noted that "**The most commonly reported issues discussed by students attending the National Fire Academy, NFIRS Program Management class are: the need for data analysis tools and solutions for quality control and assurance issues at the state and department levels, regardless of the software packages being used.**"

My Fire Rules was initially developed to address the second of these problems with our **Incident Validation**; our more recent software offering, **Incident & Patient Metrics Toolkit** (a set of "Pivot Tables in-a-box") now address the first problem. Realistically, reporting on bad data will give bad answers, so in effect the Fire Service must deal with the number 2 problem in order to have any hope of obtaining any meaningful reporting of NFIRS Data. The functionality of *My Fire Rules* **Incident Validation** has been tailored to assist the local "FireRMS Administrator" to "enforce Business Logic Rules down to the Data Entry level". This is accomplished by providing a "Rules Definition environment" that distills the creation of "Rules" (SQL Select statements) that locate "bad data" down to the creation of a SQL WHERE clause. The location of this "bad data" is the key to enabling you, the FireRMS Administrator, to enforce your "Business Logic" onto the data entry of Fire & EMS Reports. These Rules can be evaluated when a user is finalizing his or her report with the use of our included **My Validator** program, a very small multi-user exe that presents a report of any errors along with the location within the Incident Program to go fix the error(s). In RescueNet FireRMS (from ZOLL Data) there is a "hidden" button that allows products like *My Fire Rules* to present a report to the user where the results of these Rules about "bad data" can be displayed in a meaningful way allowing them to go back and correct their data entry errors immediately. If one or more of these Rules find "bad data" in the Incident Report that is being completed, these Rules can be enforced to prevent the user from saving the report as "Complete", even if they "mark the report as complete", it will be "un-marked" as they close the Incident Report. Most end users want to report meaningful data, but with the vast scope of NFIRS data elements, combined with the increasing complexity and training requirements of the real job of being a Firefighter/EMT, perfect reports are more than mere mortals can be expected to produce without a tool like *My Fire Rules*. To further assist end users in understanding what (and where) the problems are in their reports, the text of the "Problem Message" (what is presented to the end user in the **Validation Report**) is fully definable by you, the FireRMS Administrator. This provides for complete localization of the Rules so that the intent can be fully understood by the end user.

My Fire Rules fills a major void that has existed in the Recording of Fire and EMS Incident Data; how to encourage and enable Fire and EMS personnel to capture useful and complete data.

My Fire Rules **Incident & Patient Metrics Toolkit** (part of our "Pivot Tables in-a-box" solutions) provides the tools that allow anyone to delve deeply into the Incident & Patient data (minus names or identifying data) very rapidly and efficiently. Our Pivot Tables are provided in an Excel Workbook that: we seamlessly connect to your FireRMS Database allowing for almost immediate inclusion of data as soon as it is entered, pre-calculates commonly required data points, has an intuitively prepared "Pivot Table Layout" making sense out of the data more easily, provides fully defined "Code Description" fields allowing understanding of the data without having to "know the codes",allows for "Drilling-Through" to the underlying data to easily examine what the "Summary Data" is telling you, Pivot Tables provide the avenue to ask those pesky "What about…" or "What if…" types of questions; they allow the user to ask virtually any question of your data and get the results immediately.


My Fire Rules, My ePCR Metrics & My Fire Touch
from The Rules Guys LLC
Start with ZOLL's FireRMS & ePCR
apps for fire departments & EMS agencies
to get your SQL Database & Basic Reporting
SQL Database
Basic Reporting
SQL Data Warehouse
Add our configuration apps...
MyFireRules & ePCR Toolkits
to configure your SQL Data Warehouse
Add our productivity apps...
My Validator for real-time FireRMS data validation
Metrics Toolkit for FireRMS data analytics
ePCR Metrics for ePCR data analytics
Recovery Claims & Billing Toolkits
for FireRMS revenue generating solutions
Add our My Fire Touch apps
for dashboards & mapping
on your mobile devices

Why SGL?

After talking to many VFP developers I came to the realization that they would not consider using Xbase++ until it had development tools similar to VFP, especially for GUI screen design. They had been introduced to Xbase Parts and had seen a lot of finished Xbase++ applications that were impressive in their GUI capabilites, but that did not convince them that it could work for them. But I was convinced that it **would** work for them. Why? Because I had sold hundreds of my eXpress++ product to Clipper, dBase and FoxPro programmers who had no experience at all with GUI programming and most of them created spectacular applications with great visual appeal. So I made an appeal to the VFP developers who attended my session to let me prove it by giving me an application to convert. I said that I would charge my regular hourly rate and that they would be pleasantly surprised at how quickly and inexpensively this would be accomplished. SGL was presented at the Xbase++ conference in Frankfurt in 2015. The white paper for that session can be viewed on the Donnay Software website at http://donnay-software.com/ds/Donnay_SGL_Frankfurter.Pdf.

Migration Strategy

When deciding on a migration strategy, I must first be convinced that there is nothing in the VFP application that cannot be accomplished by Xbase++. The MyValidator application consists of several GUI screens that include pushbuttons, tab pages, labels, checkboxes, combo boxes, array browses, single-line edits, multi-line edits, html viewer (ie activex control), radio buttons, group boxes, spin buttons, bitmaps, progress bars, etc. Xbase++ supports all of these common controls therefore I determined rather quickly that creating the screens would be simple and straightforward.

Screen Design

Xbase++ does not include a screen designer, however it does include a robust set of GUI controls which are referred to as Xbase Parts. Back in 1998, when I first started working with Xbase++, it became obvious to me that, without a screen designer, there was a need for a language extension to Xbase++ that would simplify the design of screens because using native Xbase Parts code was too time consuming and too difficult to maintain. This is what inspired me to create a SGL (Structured GUI Language).

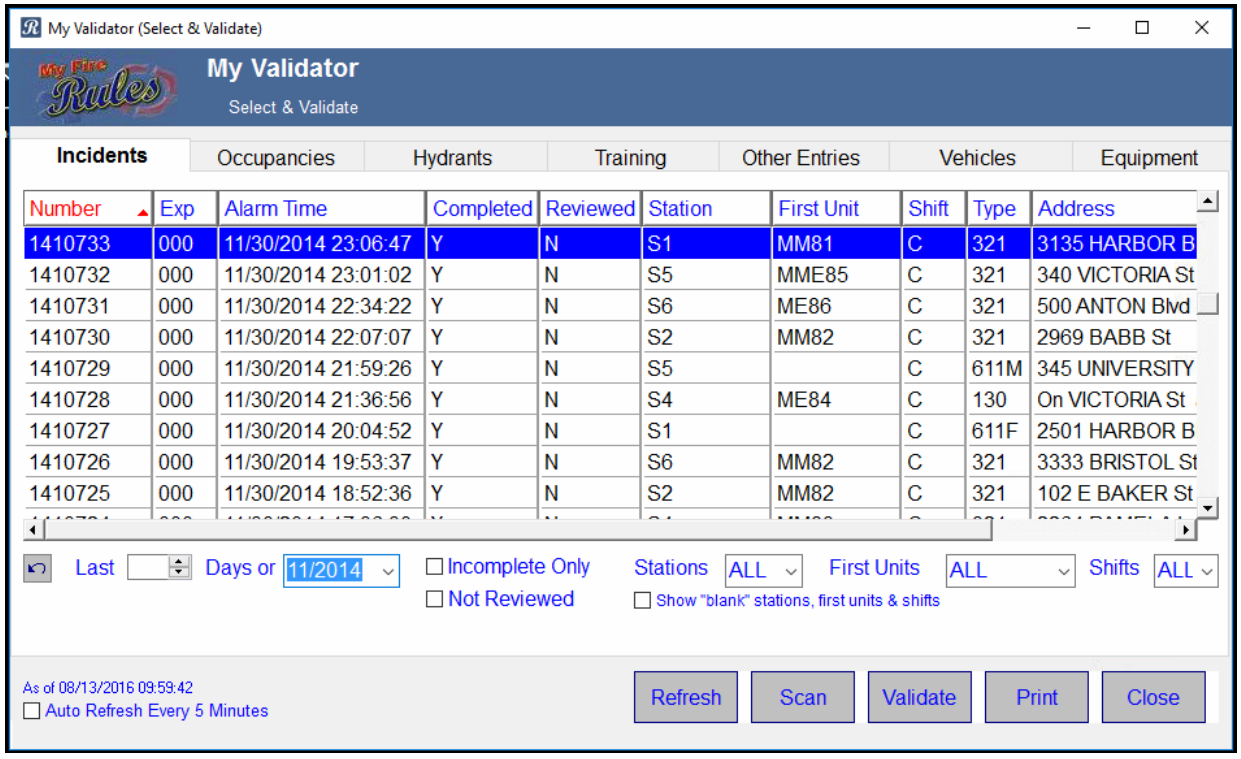
Several times, in the past few years, I attempted to write a conversion tool that would take the GUI (forms) definitions from the VFP data and convert it to Xbase++ code. That proved to be a monumental task which could never be realized. I then tried to convert the VFP form data to eXpress++ SGL code. That was much easier, but I still realized that it would be a not-worthwhile undertaking. Basically, my goal was to allow VFP developers to continue to use their Microsoft development environment to create and manage screens and then convert the result to Xbase++ code. I will not go into the many reasons why this was a bad idea. Instead, I will explain the strategy I used to convert the screens (VFP forms) to SGL code.

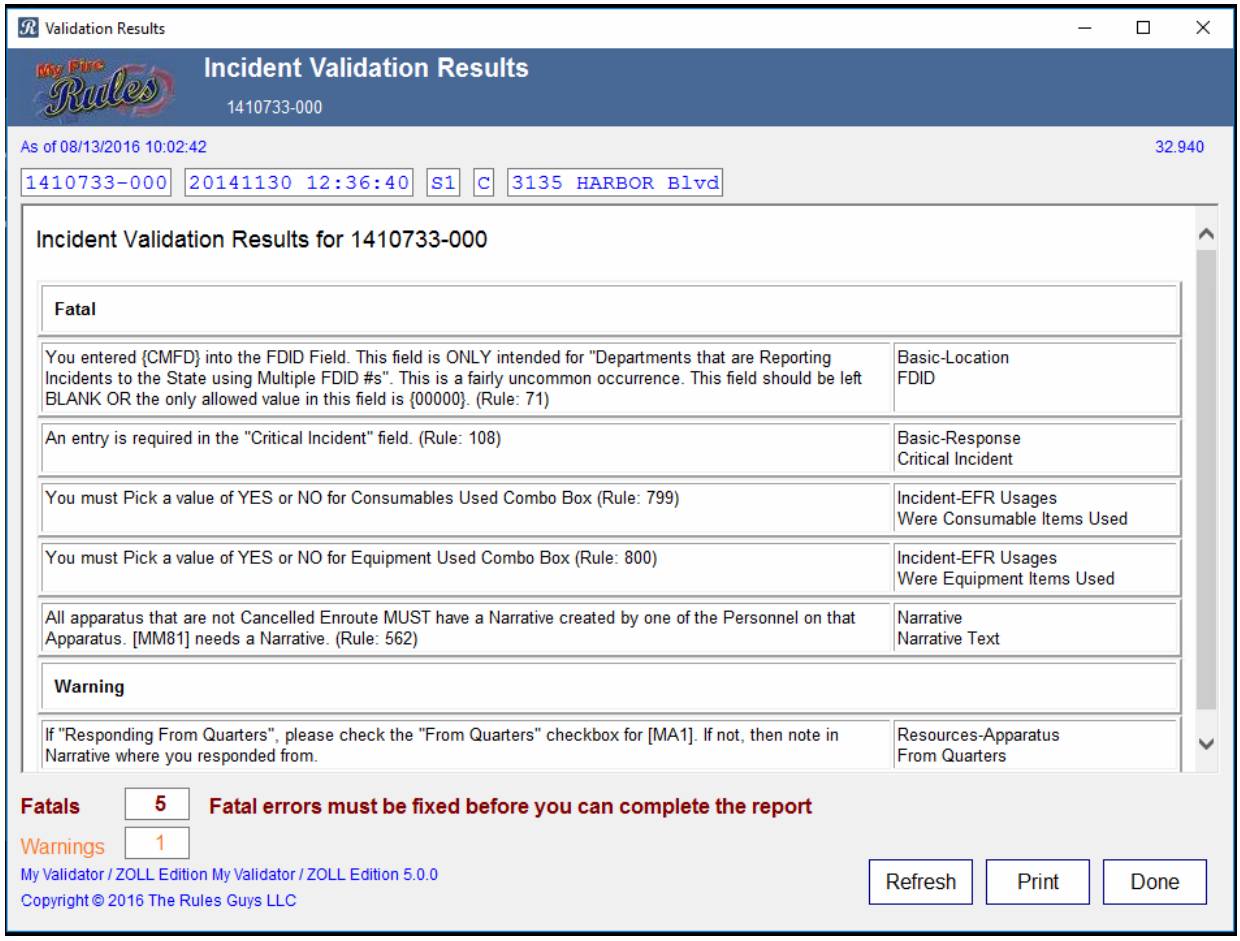
I have learned, over many years of working with other applications, that the one thing that is common between different languages is the look and behavior of the screens. The underlying code may be entirely different, but the functionality tends to follow a standard set of rules for Windows-based desktop applications. GUI elements and menus all behave the same whether the application is written in VFP, Delphi, Xbase++, VB, C#-dot net, etc. Based on that insight, I learned that it is much faster to write the screens entirely from scratch, using a declarative language, rather than to try to migrate the code. This is what I did with the MyValidator application. Not one time did I look at any of the VFP code or run the VFP

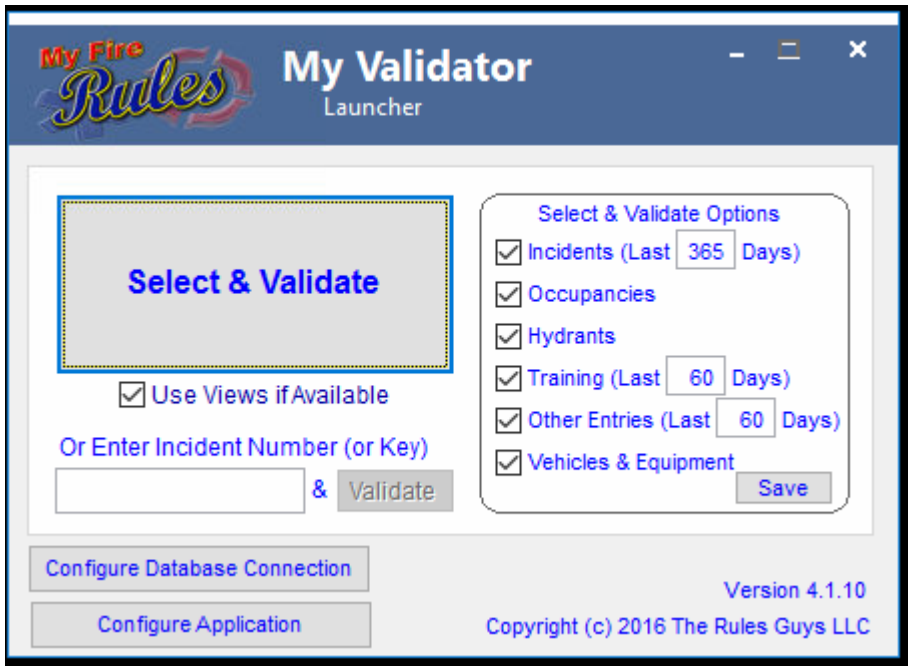
studio. My screen designs were simply built by running the VFP application and observing the look and behavior of the screens. I had to do this in 2 steps:

1. Create the GUI elements.
2. Connect the data and functionality to each GUI element.

The first step was actually the easiest and quickest because I only had to create the same visual look. Not one time, during this process, did I have to consult with the VFP developers. After that was completed, it was then that I spent only a few hours of phone discussion with Stewart about how the application was supposed to work. TRG set up a virtual machine in which the original application was installed, including the SQL server. I used TeamViewer to connect to that machine and I uploaded the Xbase++ development tools and eXpress++ development tools. During our consulting conferences, we both connected to the remote virtual machine. The development of the screens took about 20 hours. Below are an example of what some of the screens (forms) look like.




Validation Results
Incident Validation Results
1410733-000
As of 08/13/2016 10:02:42
32.940
1410733-000
20141130 12:36:40
S1
C
3135 HARBOR Blvd
Incident Validation Results for 1410733-000
Fatal
You entered {CMFD} into the FDID Field. This field is ONLY intended for "Departments that are Reporting Incidents to the State using Multiple FDID #s". This is a fairly uncommon occurrence. This field should be left BLANK OR the only allowed value in this field is {00000}. (Rule: 71)
Basic-Location
FDID
An entry is required in the "Critical Incident" field. (Rule: 108)
Basic-Response
Critical Incident
You must Pick a value of YES or NO for Consumables Used Combo Box (Rule: 799)
Incident-EFR Usages
Were Consumable Items Used
You must Pick a value of YES or NO for Equipment Used Combo Box (Rule: 800)
Incident-EFR Usages
Were Equipment Items Used
All apparatus that are not Cancelled Enroute MUST have a Narrative created by one of the Personnel on that Apparatus. [MM81] needs a Narrative. (Rule: 562)
Narrative
Narrative Text
Warning
If "Responding From Quarters", please check the "From Quarters" checkbox for [MA1]. If not, then note in Narrative where you responded from.
Resources-Apparatus
From Quarters
Fatals
5
Fatal errors must be fixed before you can complete the report
Warnings
1
My Validator / ZOLL Edition My Validator / ZOLL Edition 5.0.0
Copyright © 2016 The Rules Guys LLC
Refresh
Print
Done


My Validator
Launcher
Select & Validate
Use Views if Available
Or Enter Incident Number (or Key)
&
Validate
Select & Validate Options
Incidents (Last 365 Days)
Occupancies
Hydrants
Training (Last 60 Days)
Other Entries (Last 60 Days)
Vehicles & Equipment
Save
Configure Database Connection
Configure Application
Version 4.1.10
Copyright (c) 2016 The Rules Guys LLC

Business Logic (the code)

When writing the functional code, I basically emulated the functionality that was described by Stewart, some verbally, some in writing. I did not need to see any of the VFP code to do this, because it was not difficult to write the underlying code that connected the data to the controls. A VFP developer who knows the FoxPro language will find many similarites in the Xbase++ language, with the exception that Xbase++ introduces some new features that actually can simpilify the code. If you are a VFP developer who is contemplating converting your application to Xbase++, I suggest that you first read my white paper: http://donnay-software.com/ds/Donnay_Xbase_for_VPF_Frankfurter.Pdf.

The MyValidator application is rather unique in my experience as compared to the many Clipper, dBase and FoxPro applications I have worked on over the years. What makes it unique is the fact that most of the intellectual property is in proprietary stored procedures using Microsoft's Transact SQL. Due to the fact that SQL stored procedures are not specific to any Windows language, they can be used by both the VFP application and the Xbase++ application, even concurrently. This is because both languages can connect to SQL databases via ODBC. Xbase++ includes a data driver called the ODBCDBE. This DBE allows connection to SQL data via the same connection string that is used in the VFP application. Both languages support mechanisms to use the returned data as a database in a workarea.

The Database

The VFP application does not use any databases other than the SQL data that is accessed via the ODBCDBE. Probably the most amount of time expended on this conversion was due to small anomolies and differences in how the ODBCDBE calls SQLServer and how VFP calls SQLServer. In 9 out of 10 places there was no problem, but in 1 out of 10, it took some extra time to figure out why I did not get a correct result. To be clear, Xbase++ did not return ambiguous results. On the contrary, when it did get a result it was always 100% accurate. The problems were related to issues in which Xbase++ got no result at all. Two of those issues were resolved by setting required properties of the SQL server session object. Another two of those issues required that the VFP developer make a small change to one of their stored procedures and another change to the password format for the connection.

Enhancements

The Xbase++ version of MyValidator performs faster than the VFP version when it comes to the actual execution of the application code due to the fact that Xbase++ is a faster compiler. It's performance is equal to the VFP version in areas where SQL SELECT statements or EXECUTE sp statements are called because this logic is perfomed on the SQL server.

The Xbase++ version has some multi-threading which allows multiple windows to be opened. It behaves more like an MDI application than an SDI application.

The Xbase++ code is easy to maintain because the entire application exists in one source (.PRG) file.
All screens and code are in this file as compared to the VFP application which exists in many different files, few of which can simply be viewed or edited by a source editor. The MyValidator source code consists of over 100 files. Each screen (form) is contained in a set of data files which can only be opened by the VFP studio. The Xbase++ code can be maintained by any text editor or by the Xbase++ workbench.

The source code will be shown to attendees of the conference but is not included in this session paper nor included in the conference materials for reasons of proprietary nature. Also, both applications (the VFP version and the Xbase++ version) will be run to show functional and visual comparisions.