

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2007-10/msg00565.html>

- *From:* Robert <no@xxxxxx>
 - *Date:* Wed, 17 Oct 2007 00:05:29 -0500
-

On Tue, 16 Oct 2007 23:09:26 +1300, "news" <greg@xxxxxxxxxxxxxxxxxx> wrote:

"Robert" <no@xxxxxx> wrote in message
news:7hh8h358vkj137oecabhvd5ausrqp2kgj@xxxxxxxxxx

On Tue, 16 Oct 2007 11:32:09 +1300, "news" <greg@xxxxxxxxxxxxxxxxxx>
wrote:

I'm looking for comment on any issues I may encounter in a conversion for Microsoft COBOL V5 (Which was licenced off Micro Focus) and Version 5 of Net Express.

Background

I am the owner/operator of the company that owns the copyright to the source code. I am not a programmer (Although I understand the principals of programming and suprise myself at what bugs I can fix) The system is currently a PC Based Financial suite. There are between 500,000 and a million lines of code (Is that large?)

Yes, that's large.

Hmmm, thats what I Thought. I guess 20 odd years of coding makes for a bit of a mammoth conversion task.

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

There are about 100 installations of the 16 bit version of the system.
Companies using it range from NZ\$400M turnover to \$100K turnover (What New Zealand would regard as Small to Medium sized businesses)
ISAM data structure.
Current system is 16bit console based, stable and fast.
User Support for the product just won't die (They like it)
However
development constraints such as memory issues of staying under 640K and
printing issues are getting more expensive to support.
I have an Evaluation Edition (Time constrained latest full version) of Net Express and Visual studio 2005 loaded on a Windows 2003 Terminal server.
I have paid for and completed the initial phase of a proof of Concept. One Module (100,000 lines of code compiled and has produced working executables but not fully tested)

If you have one program with 100,000 lines of code, you have serious structural problems that need to be remedied before attempting 3–6 below

The suite is made up of over 200 executables which draw on over 600 copybook files for reusable code.

Now the average program size is down to 4,000 lines, which is high—average for programs written in the 1980s. That's not horrible, like 100,000 lines, but is still too big.

When they say reusable code, they mean procedural code, not data structures. Outside OO, reusable code is written as callable functions.

I'm hoping a complete restructure isn't in order or this project won't fly.

No, just repackaging. Package logical chunks of code as callable programs, with special attention of file IO and user interface. Those are the programs you're going to rewrite as database and GUI, respectively.

I now have to assess cost of several logical phases. i.e. assess the min and max cost possible for each phase – This is where any issues I

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

need to consider would be valuable, I can't research what I don't know.
On completion of this analysis I will go out to the Userbase with an indication of costs they will need to absorb.(several key players in this group have been informed of this process) If I get support, the conversion will go ahead, if not, I will create a strategy to exit the market.

Phases

1. Straight conversion to a 32bit console version of the system.
Possible consolidation of executables. No improvements. Users having memory issues would no longer get these symptoms.

Very easy. It's just a recompilation. You don't need phases.

Converting data files is just a reorg. You don't need to change data elements in your records.

2. Write a module to replace ASSIGN PRINTER to handle printed output.

Many printer drivers that can do what you want without changing source code.

My goal is if it prints from windows it also prints from my programs. However as I want to port it to linux it will have to take this into account in design.

Windows' printer interfaces are much more sophisticated than Unix's. It is very common to have the business logic on Unix and the user interface on Windows. A printer is as much a user interface device as a screen. Middleware ties the front and back ends together. Well known examples are WebSphere, J2EE, JBoss, TIBCO, Weblogic and Struts. If you like .net, use ADO.

Do not try to do it yourself with first generation approaches like RPC, or even Samba or NFS.

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

3. Refactor programs to reasonable size, one function per callable program. Separate user interface from application logic.

See above this may already be the case.

7. Port to a relational database (Probably Postgres) for easier 3rd party reporting.

I would make this 4. First, move file IO into data layers. Second, design the database .. properly (normalized), NOT one table per indexed file. Third, rewrite data layers to read the database and return 'logical views' matching indexed files one for one.

Having done 1, 3 and 4, you have a foundation to build on.

3. Write a web interface using Net Express and hopefully AJAX to replace the screen I/O of the current version. Duplication of look and feel of existing interface envisaged. Compile as a .Net application.

Possibly, but I need the users to get something in a timely fashion with a low risk of introducing bugs. This strategy would seemingly suck a lot of cash on testing before I could get the new code generating any new revenue.

Easier than you think .. AFTER user interface is separated from business logic.

4. Port to Linux using MONO (.NET framework for Linux)

Why? Unix GUI isn't as good as Windows.

Because Linux has a big chunk of the server market.

You mean Unix, not Linux. That's true, but Unix is seldom used for user interface (except on Apple). I work on very large Unix servers for F-100 companies, and I've never seen KDE

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

or GNOME used to interface with human beings, only with developers. :) The GUI world used to be owned by Windows, is now owned by Web pages written with Java tools such as J2EE.

E-commerce is expanding,

If I can provide the reliability and stability of a COBOL based financial system for e-commerce systems to write to directly I believe there will be a market for it. I can provide point of sale through to Balance Sheet on one box with one core business logic.

There are many BAD point of sale systems in the world, fewer than a dozen good ones.

I've seen elegant integrated systems that start with a balance sheet, let you drill down to accounts receivable, down to customers, down to invoices, down to line items ... all threaded together with a unified audit trail such as a keyrec number. It looks good in demos, but the real measure of utility is in the gritty details.

I don't ever intend to design a non-webform version of this software. I intend to leapfrog that technology and never offer it to the market at all. As far as I'm aware, the browsers available for linux are in many ways superior to those in windows, particularly in security. At the end of the day whilst everyone likes "Pretty" most serious business people will opt for solid and functional when trusting their financial transactions to a computer system.

Well said. By the same token, they don't judge a system by its browser security, either. Leave that to magazines.

5. Revisit the user interface (3.) altering bigger sections of underlying business logic to enable us to produce a "prettier" more flexible interface.
Possible conversion of some code to Object orientated code.

People will tell you to do the user interface with Java oriented tools

AJAX is a java orientated tool. I'm yet to figure if I can get it to work with Net Express

6. Alter Code to handle security for system to run as a web application in a bureau environment. 5. & 6. may occur simultaneously.

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

Do it yourself security is almost always third rate. Security should be outside application code, primarily in the database. That's one reason why you first need a database.

This step was not necessarily to be coded using COBOL. My thoughts were more based around Linux and its ability to secure the ISAM data files as part of the registration process of signing up to the web service. This may or may not be done via COBOL. The code I envisaged would simply look to a different directory for the datafiles depending on which user was logged in.

ISAM data files!! Be serious. A schema is the database equivalent of a directory; a table is equivalent to a file. Databases have security not only at those two levels, but also at the individual field within a record. Some users can see salaries, others cannot. Some can update salaries, others cannot. With security in the database, rather than application code, no one can bypass it by using general purpose tools nor by writing a program.

Programmer Strategy

1. Devide the tasks into managable portions.
2. Clearly define deliverables for each portion.
3. Go to Global tender for each Task using probably using outsourcing websites (payment on deliverables or agreed milestones).
4. Coding done using my Terminal server.
5. Manage coding pactice, the meeting of timeframes, quality of deliverables, documentation practices etc, using VoIP, video conferencing, and remote control of RDP sessions, personally.

You overlooked testing, which will be more expensive than coding. You overlooked design, as well.

Typically I do that bit and pay myself stuff all. :(

Development plans emphasize the things a company is good at and/or believes important, and deemphasize the others. The development plan DOES NOT CHANGE REALITY. By reducing testing to a single line item and giving it a small time/money budget doesn't reduce the need for testing, it means you'll deliver a poorly tested system. Some companies go the other way. They spend 90% of their time and money on planning, process control and testing, while sweeping development under the rug. They wind up with software that's technically weak, but well documented.

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

Re: Strategy for (Large?) 16bit COBOL code conversion to Net Express

Where do you want the errors to be? They will be found wherever you 'saved money' or couldn't be bothered. If you want high-level design errors, design on the back of an envelope.

Anyone interested at this level is welcome to make contact.
Please be
aware
that without support of the userbase that this project will be
all over
within the next 6-8 weeks as I won't have justified the
purchase of the
development tools.

Show them mockups of pretty GUI screens with searchable lists and reports
as spreadsheets.
They'll fall in love.

They are so used to the console interface that many have threatened me with death if I change it. :)

We are talking of Java enabled Cellphones sending XML data to the linux server (I mentioned earlier) to load Payroll transactions and automated SMS messaging from that server to communicate job locations and descriptions organising casual remote workers. That seemingly has them excited if not in love! Can't really be done without new tools!

Apparently you think CCXML and VoiceXML is the best or only way to communicate with mobile users.

Generally, there are two competing network technologies: telephony and computer (IP). The XML ones originated in the telephony side (bet you didn't know that). As a result, they depend on centralized control (SS7, telco switches) and metered pay per use, because that's how telephone people think.

Competing technologies from the computer side are SIP/RTP and I.323. That's how VoIP phones work. At a low level they're carried as UDP packets, usually on the internet or VPN. Both can send text messages, voice and video to any COMPUTER, telephone or IP-capable speaker or screen. Your users do have computers. The phone company doesn't want you talking to computers.

An emerging technology, independent of those those two, is Jabber/XMPP, which is pure internet. Google has added significant functionality to the Standard with its own flavor, called Jingle. Don't laugh. Google is serious, and we know they'll own the world in 20 years. Google offers a FREE library that can dance circles around SIP/RTP, talking over the FREE internet.

When the telephone company salesman tells you "XML will do for voice what HTML did for screens" he's just selling phone services. The computer-based approaches are more

Re: Stategy for (Large?) 16bit COBOL code conversion to Net Express

liberating.