

## ***For a Few Dollars Less***

### **Migrating FlyBuys from MicroFocus to Fujitsu COBOL**

**Tuesday 02 February 2010**

Are your software systems written using MicroFocus COBOL?  
Are you finding COBOL compiler licensing costs exorbitant?  
Want to find out what's involved in converting your software from using MicroFocus COBOL to Fujitsu COBOL compilers?

Then this article is for you!

Early in 2009, Shine Technologies performed a conversion of the COBOL source code used in FlyBuys systems to replace the use of the MicroFocus compiler with the Fujitsu COBOL compiler and related runtime libraries.

Read on for a high level overview of the activities involved in this project, and how it ultimately turned out.

#### **Primary Aim**

Our primary aim was clear from the beginning – remove all traces of MicroFocus COBOL compiler and runtime library packages, and replace with the equivalent Fujitsu COBOL offering.

Or to put in another way:

Convert production FlyBuys systems to use Fujitsu software for compilation in as short a time as possible, and ensure that they work correctly.

#### **Why Would You Want To Do This?**

When speaking about this project with peers, I'm invariably asked: "Why migrate from COBOL to COBOL? Why not rewrite the system using (say) Java?"

An excellent question. The answer is a tale of "time and money".

Rewriting the existing COBOL software using Java was not possible given the timeframes we were given (described later in this article).

To explain further, the client was in a difficult position. For particular business reasons the client needed fast migration.

The fastest way to achieve this would be to “simply” use another COBOL compiler, which did not have the same restrictions.

As much as personally I'd love to convert the whole FlyBuys COBOL system to Java in one big bang, we had to remain true to our philosophy of *delivering business benefit over applying new technology* (See the Shine Philosophy at <http://www.shinetech.com/about-us/our-philosophy>).

And so, our “For a Few (a Lot of?) Dollars Less” our adventures began...

### **Some Secondary Aims**

At the start of this project, we took a step back and had a long think about what we were about to embark upon.

We would be changing every single COBOL source file, as well as many of the other software that was closely related or integrated with COBOL executables (such as C programs and libraries, perl, shell and even SQL scripts).

As we were going to be changing “everything”, we decided to add a few very quick and easy enhancements around the fringes to make the system better, in various ways:

1. We added source code identification into every source code file. This enabled us to uniquely identify the source code and versions that comprise of any single executable or script. Easy to do, and invaluable for software release and bug tracking activities.
2. We recognised that we would have a need to rapidly promote large numbers of executables through our build, system test and UAT environments at a moments notice. So we heavily reworked our build scripts, and used the Hudson extensible continuous integration server to harness our software builds and promotions to different environments. This worked like a charm. To quote one young developer “Hey, I just clicked a button and deployed only the 28 executables that I needed to.” 'Nuff said!

### **The Code Base Impacted**

The entire FlyBuys software system consists of several types of technologies, operating on a combination of Linux and HP-UX operating systems.

The bulk of the software is actually written in Java, with new or significantly changed functionality always being delivered using Java. However a large part remains which is written in COBOL.

The “COBOL” part of the system forms a part of the core backbone of back-end batch processing, as well as providing the Call Centre with screens for handling customer enquiries and updating details to the back-end database, and is crucial to the business.

In theory, we were just changing COBOL source code. The reality was somewhat different.



Changing the COBOL compiler used has an insidious flow on effect. Almost anything that interacts with COBOL, either during software compile or during runtime needed changing - sometimes in a minor way, sometimes not.

A few examples are:

- Scripts (perl and shell) which would execute COBOL programs had a particular way of specifying output file names for the COBOL programs
- SQL scripts which were used by COBOL programs would have a different way of being used
- C library routines which were used by COBOL programs would have to change (called differently, return values differently)
- COBOL functions that were called from C library routines would also have to change (called differently, return values differently)
- COBOL screen/form user interface programs would use a completely different terminal key mapping mechanism, so parts of the Operating System environment would have to change
- The Operating System environment would also change due to different environment variable names and concepts being used

*Note:* COBOL software is also integrated with the newer Java software, using sockets for communications between legacy COBOL servers and new Java servers. It must be noted that in these situations no code changes were required in either the COBOL or the Java source code. The newly compiled Fujitsu COBOL software integrated and communicated correctly with Java servers without issue.

To give an indication of the code base that was impacted, a count of related source code, executables and related artefacts follows.

COBOL Counts:

- 918 COBOL source code files
- 190 COBOL executables

C Code Counts:

- 141 C source code files
- 2 C libraries
- 30 C executables

Scripts requiring changes:

- 100 perl scripts required minor changes
- 30 sql scripts required minor changes
- 6 shell scripts required minor changes

## Project Numbers

We had **6 weeks** to convert the system, including full rollout to production.

Our personnel at the coalface consisted of...

**NOT** 50 developers,

**NOT** 20 developers,

**NOT EVEN** 6 developers, but...

Wait for it... 4!

*Four (4) developers to convert a large system in 6 weeks and install into the production environment.*

Four **TITANS** I say!

## The Plan

After understanding what was required, in terms of timeframes, personnel and scope, (and regaining consciousness ;-), it was time to come up with an action plan.

Broadly, this consisted of the following actions, each with their own milestones:

1. Backup and Remove MicroFocus build tools and compiled executables
2. Install and configure Fujitsu Compile and runtime tools
3. Bulk Convert all source code as much as possible, and compile using Fujitsu compiler (flag those requiring by hand conversion) – we rolled our own conversion tools using perl, awk and sed, which could be re-run as required
4. By Hand convert all source code and compile using Fujitsu compiler – developer intensive effort to replace or create functionality that existed with MicroFocus, but did not exist in Fujitsu COBOL
5. At this point all COBOL, C and scripts have been fully compiled and ready for testing
6. Perform Run Testing and Fix-ups – ensure that at a glance the software performs as it should
7. Perform System/UAT Testing – the client tested to ensure that functionality had been preserved
8. “Big Bang” Production Rollout – all new software rolled out at a particular point in time. Done this way because it is not feasible to operate MicroFocus and Fujitsu COBOL build/run software in the same environment

## Types of Problems

The types of problems that we had to solve in converting the software fall into a few categories.

**Problem:** Keyword and function name changes between MicroFocus and Fujitsu.

**Solution:** We rolled our own bulk edit scripts, which would make simple keywords and functions changes from MicroFocus to Fujitsu versions of same.

**Problem:** MicroFocus COBOL functions which had no direct Fujitsu equivalent.

**Solution:** Re-write COBOL by hand so that the program would operate correctly.

**Problem:** No equivalent COBOL runtime library function available in Fujitsu.

**Solution:** Write our own function in either C or COBOL and integrate into the COBOL software.

**Problem:** COBOL screen/form based programs did not have input field termination. They could not detect when a field had been changed.

**Solution:** Turned out this was a genuine bug. Fujitsu provided a patch to their software fairly quickly, and this problem was solved.

**Problem:** COBOL programs populating database tables with garbled data, and giving no indication of problem.

**Solution:** Wrote our own tools to identify of offending data types and techniques, and changed software appropriately.

There are many, many specific variations of these types of problems that we encountered and addressed.

### **So... Where Are We Now?**

Summer has turned to autumn, autumn to winter, winter to spring and spring turned to summer again, to complete the circle.

A year has passed. The COBOL conversion project has long since been completed.

Was it all worth it?

YES!

The FlyBuys COBOL systems have been running well on Fujitsu compiled software for a long while.

The business was freed to pursue a new business model. They successfully launched their new affiliates program, with several affiliates joining to date.

*"The challenges were many, the expectations were high, and the timeframes were critical... Shine delivered on all fronts.*

*Shine was involved in the Fujitsu COBOL conversion project from conception to implementation and right through every phase, the team projected confidence in their ability to deliver and were never fazed at any of the obstacles. The feasibility study, project planning, development, unit testing and implementation were performed with a high degree of professionalism. Most importantly the project was completed on time, with minimal disruptions to daily operations. This is a unique achievement and to be honest, the Shine team exceeded our expectations, by far. A very challenging and rewarding experience for both sides, and FlyBuys continue to reap the benefits."*

**- Marlon Coelho, FlyBuys IT & T Manager**

End users of the impacted software at the coalface have long since become used to the minor differences in the software they use on a day-to-day basis.

Some of our Secondary Aims have also proven worthwhile, with major improvements in the ways we identify, package and deploy FlyBuys software today. From this perspective, the system is better than it ever has been before.

And, as happens, we performed the same type of conversion project all over again for systems used by another leading Australian retailing loyalty program. But it was easier the second time 'round... we just did it and went home!



## For Your Consideration

Looking back, we learnt a lot of lessons for this sort of project. In no particular order, these are things that you may want to consider or expect if you're going to perform a similar undertaking.

Your software will not “just compile” with the new compiler you're using without modification. There will always have to be some effort involved in modifying your base source code.

You need full buy in from the client and end users. In our case, the client was brilliant. They fully understood how crucial this conversion was to their business, and went out of their way to support it as much as possible. This included temporarily suspending other projects, agreeing to a software code freeze, providing personnel to perform testing, and understanding that the new system would differ in minor ways from the old one.

Expect to convert everything yourself – even if conversion software is provided.

COBOL source is not the only thing that will change. There is a flow-on effect to other related software.

There is no such thing as a “spurious” error – during a compile or execution of a program.

Don't get sidetracked trying to make functionality improvements as you go. Focus on the conversion, not a better system. Otherwise you'll never finish!

Expect to have to write/create new code to replace functionality that “just worked” before – e.g. screen field navigation.

There is no workaround for some problems – it could actually be a bug (if it just makes no sense), or you'll have to find a workaround, or possibly do without a particular functionality.

## The Soft Pitch

If you're in the same position as FlyBuys was about a year ago, there is hope.

They turned it around, moving away from restrictions that were holding them back, so it's possible for anyone to do so.

If you need to talk to Shine Technologies about this sort of work, don't hesitate to call us. Our contact details can be found at <http://www.shinetech.com/contact-us>

Good luck, and I hope that you have as much fun and success as we did!



Shine Technologies is a boutique IT consultancy that has forged an enviable reputation for delivering real value to clients. Formed in 1997 in Melbourne, Australia, Shine's collective vision is to be recognised as the best enterprise software development house in Australia as recognised by our clients and staff. Shine has deep technical skills in:

- \* Mobile Development including iPhone, J2ME and mobile optimised web applications
- \* Java Technologies
- \* Agile Development
- \* Energy industry
- \* High Volume/High Processing/Distribution Systems
- \* High Volume Web Development
- \* Complex, High Risk Projects
- \* Content Management Systems

Key clients include FlyBuys, MYER one, Sensis, ANZ Bank, NAB, Lufthansa Technik, Hutchison, Jackgreen Energy, AGL, Country Energy, TRUenergy, Integral Energy, Origin Energy, Aurora Energy, Energex and Ergon Energy.

Shine Technologies has developed several world-class solutions including NBV™ and NBM™ for network billing reconciliation in the Australian Utilities industry. Shine's NBV™ system was awarded the iAward for *"the most innovative financial application in Australia"*.

For further information visit [www.shinetech.com](http://www.shinetech.com) or contact Mark Johnson, Principal, ph: 0418 144 966

Ends.

