Richard C. Steiner

(Phone: 952-947-0438 e-mail: rsteiner@visi.com)

Resume Addendum II:

WorldFlight System Overview/Technical Background

Most of my experience at Northwest Airlines was obtained as a member of the technical team (approximately 35 people at the start in 1988, but reduced after the economic downturn in 1992 to a group of roughly 12 to 14 people) that did all of the development and support programming work related to the "WorldFlight" system, a large Unisys 2200-series mainframe-based transaction system which was the central or "core" computer system used by both the flight operations and the ground operations personnel at NWA, as well as the people in the SOC (shorthand for "System Operations Control", the operational nerve center for the airline).

As a member of the WorldFlight team, I spent most of my time performing the following types of activities:

- Software and database design work (both individually and collaboratively with people from the SOC).
- Software development (both individually and as part of two- and three-person programming teams).
- Support programming (finding and fixing production problems, making changes to the system due to operational changes or FAA mandates, making infrastructure improvements, etc.).
- Creating or updating technical documentation (all WorldFlight program modules and files were required to have a dedicated technical document which described in great detail its interfaces and operation [in the case of programs] and both its logical and physical record layout [in the case of files]).
- Being on the 24x7 on-call support team (which included answering questions asked by the users and providing hands-on solutions for live production problems as they occurred in the system).

The WorldFlight system was arguably one of the most "critical" systems that existed at NWA. It was based on an older Unisys applications technology (the original system, called UNIMATIC, was developed at United Airlines starting around 1967 on a UNIVAC 1108 and obtained by NWA in 1988), but the system was tremendously stable given the large and varied amount of activity that occurred on it from day to day. On the few occasions when the system was down, the airline took a lot of flight delays and ended up losing a lot of money. Even individual application glitches could result in flight cancellations or delays due to a lack of critical information. One of the few system outages I remember (caused when a fiber line was accidentally cut a few years ago, isolating the SOC in Building F from the WorldFlight system in Building J) even made the papers here in the Twin Cities because of the outage's impact on the airline's daily operations. Quite frankly, I felt that it was quite an honor to be one of the programmers supporting such an important system.

WorldFlight not only provided a large and varied set of interactive text-based applications that were used by many different employee groups at the airline, but it also automatically processed a very large number of message-driven "datafeeds" coming into the system from various sources (some inside the airline and some not) including weather alerts and other weather bulletins from the National Weather Service and other sources, passenger and bag count messages from the WorldSpan (PARS) reservations system, and both freight and cargo container information from the Cardinal (USAS*CGO) system.

The application code resident on the WorldFlight system also performed a number of other tasks, including:

- Handling both interactive and automated traffic going to and from the radio-based ACARS text terminals that were installed in the cockpit of all NWA aircraft,
- Acting as the application back-end for some of the newer Solaris- and Mac-based application displays which were being developed for NWA System Operations Control (the SOC),
- Acting as the central repository for all NWA aircraft scheduling and flight information (and as the primary system in which live changes were made to flight status and flight schedules) and also providing live flight

information to all of the other systems at NWA via outgoing datafeeds (including the WorldSpan reservations system and all of the flight status displays at the various airports),

- Performing the "weight and balance" calculations used by both the load planners in Central Load Control down in Memphis and by Ground Ops personnel at the gates at various airports,
- Performing the optimal thrust, flap, and takeoff/landing gross weight calculations required by flight dispatch in the SOC, and uplinking that information to the pilots via ACARS if the planes had already left the gate.
- Providing flight crews with both their pre-flight paperwork (flight plans, weather, optimal flap and thrust data, etc.) via printers at the various gates, and any updates they needed via ACARS while the flight was in the air.

As you can see, WorldFlight did a lot of different things in a variety of different ways, and many of the applications resident on the WorldFlight system were important to the operation. The critical nature of the system was made a little more challenging by the "hidden" or automatic nature of much of its processing – most of the processing of incoming and outgoing datafeeds was done completely without human intervention, and tracing problems in such an automated system was quite challenging, particularly when there were several different systems passing the data back and forth in real-time, and when WorldFlight environment had no real "debugger" as such at all, just traces.

The software technology used in WorldFlight was old, at least on the application layer, but the hardware it ran on was not, and the environment ran on top of OS2200, a mature operating system native to the Unisys 2200-series mainframe line. Most of the 2000 or so programs and 1500 subroutines that composed WorldFlight (roughly two million lines of code in total) were written in an older variant of Fortran, with some being written as newer Fortran modules or as ASM (assembly) routines. Most individual operations on the system were fast – a transaction (generally defined as a user keyin at a terminal followed by a resulting display screen) was generally expected to take a half-second or less to complete, and some of those keyins were able to perform several dozen I/O's while doing so. The overall system was very fast.

Software was created in close collaboration with the user representatives in the SOC. Most of the folks I worked with when doing either design work or testing were flight dispatchers, pilots, ground ops people, or folks related to NWA's meteorology department. I also did some work with other groups from time to time as projects required.

During my two years as a Unisys contractor with the WorldFlight group and my eight years as an NWA employee, I worked on perhaps 30 major projects and 100 or so smaller ones. The sheer variety which is present in this type of application programming work is very hard to summarize in a few pages of text, or in a resume, and in a system where error-free operation is absolutely critical, almost everything one does ends up becoming very important.

Most of the work that I did while at NWA was in the following areas of the WorldFlight system:

- **Surface Weather system**, including work on the code which performed reception, parsing, validation, storage, and automatic dissemination of various weather bulletins from the National Weather Service, the creation of the NWA "Turbulence Plot" alerting system for pilots, the creation of the TWIP/Microburst weather alerting system for pilots, and work related to interactive ACARS weather displays for the flight crews while in flight.
- **ACARS subsystem** (which both sent messages to and processed messages from the small text-based ACARS terminals that were installed in the cockpits of all of NWA's aircraft). Specific projects included work on the ACARS processing for Fuel On Board validation, the automatic and pilot-solicited sending of weather data and alerts to the aircraft, and the sending of optimal weight/flap/thrust data to the aircraft.
- **MGL** (also called "Gross Weights") **subsystem**. This included the code that determined the optimal engine thrust, flap/overspeed settings, and the various weight limits of the aircraft just before takeoff and just before landing, as well as the various interfaces between MGL on WorldFlight and the Unix-based Ops Database which stored historical data from the Worldflight System.
- **Programmer utilities and infrastructure**. As a support programmer in the WorldFlight group who had a somewhat more technical background than some of my teammates, I spent a lot of time working on parts of the application infrastructure. Major projects included the complete rewriting of the main display routines for all multi-page text displays on WorldFlight (added searching, arbitrary line offsets, etc.), the conversion of the octal system error printer to an online database complete with sophisticated search capabilities and both error

decoding and reporting software, the creation of a utility to help programmers from stepping on each others toes while working on the same program modules (WorldFlight had no sccs-like change management system), and the creation of a precompiler which determined and optionally reduced program resource usage for large programs, solving a huge time bottleneck when programs using the older Fortran compiler got too large. I also obtained, installed, maintained, and heavily modified a number of OS2200 programmer utilities from various sources including UEDIT (a fullscreen programmer's text editor) and FINDREF, a fancy front-end to the source code reference utilities we used at NWA (TeamQuests's CULL and IACULL), and similar in many ways to the "cscope" utility found in Solaris but somewhat superior in functionality.

Some of the projects I worked on involved the design of new databases and program flows and the creation of completely new code, others required the modification or complete rewriting/replacement of existing code. Some projects were quite large by our standards (the Flex Thrust project was perhaps 5000 hours in total, of which I accounted for roughly half, and the SYSERR database took perhaps 1800 hours to complete as a solo project).

I also spent some time (roughly a year) supporting a set of daemons written in C and running under Solaris that accepted various bits of flight and gross weights data from the WorldFlight system and stuffed it into a Sybase database using CT-Lib functions. That gave me some additional exposure to C in a Unix environment (added to the playing I'd done at home under Linux with C, curses, and gdb front-ended by the DDD debugger).

Instead of trying to describe the routine work in my resume, I've decided to highlight only the most significant accomplishments, and to show only those projects which I felt were exceptional in some manner. Believe me, there's a lot more!

I hope this document provides some useful background into the nature and scope of the applications on which I've worked. If you have any additional questions, please feel free to contact me.