Lessons from 23 years of Gypsy programming

F. Alfredo Rego
Adager Corporation
Sun Valley, Idaho 83353-3000 • *USA*

www.adager.com

A keynote message from hp3000-L

John Clogg, of Coldwater Creek, wrote a message to hp3000-L on the topic of this ***HP World 2001 Conference*** (Chicago, Illinois, U.S.A.)

I would like to take John's message as a guiding light for the three kinds of things that I wish to accomplish during this presentation:

1. Exchange ideas.
2. Give specific examples of how we can all cooperate with Hewlett-Packard.
3. Engage as many members of the hp3000 community as possible.

```
Date:         Thu, 24 May 2001 14:32:51 -0700
Reply-To:     John Clogg <jclogg@thecreek.com>
Sender:       HP-3000 Systems Discussion <hp3000-l@raven.utc.edu>
From:         John Clogg <jclogg@thecreek.com>
Subject:      Re: [HP3000-L] HPWorld cost
To:           HP3000-L@raven.utc.edu

... I haven't read Interex's charter lately, but I'll bet its
stated purpose is not only to stage profitable conferences.
There is probably something in there about "exchange of ideas"
and "advocacy with HP".  Those goals are best served by engaging
as many members of the user community as possible.
```

Always on

The theme for HP World 2001 is "Always on" and this presentation addresses this topic from the perspective of a computer programmer who has worked (and continues to work) on all corners of the globe and on all kinds of transportation vehicles while moving from one corner to another.

Integrated work and play

It turns out that (almost) everything I have done during the last 23 years has (almost) always centered around my responsibilities as Manager of Research and Development at Adager Corporation. When people ask me about my "rest and relaxation escapes" and about my "vacations to get away from it all", I am at a loss because, for me, there is no difference between "work" and "play". I love what I do and I love doing it either alone or in the company of my family and friends.

"Pervasive computing" was not always so

Nowadays, everyone has at least one computer. In the 1970s, that was not the case and I had to come up with a methodology that would allow me to develop world-class software while being based in a cobble-stoned colonial town in Guatemala with miserable telephone service.

Since 1978, I have been a guest programmer at hp3000 shops throughout the world and, in the process, I have managed to build quite a bag of tricks which I will be pleased to share with you. I will use my laptop computer and an LCD projector to present actual case studies ranging from the primitive 1970s to the thoroughly modern 2000s.

I have faced many tricky "how in the world do I do X?" questions and, with the help of many friends, I have been able to develop practical answers.

How does one develop and maintain a major software package using remote-access technologies (ranging from old and slow telephone connections to new and fast Internet connections)?

Remote-access technologies

- Hard-wired RS-232 (in the old days).
- Sneakernet (connecting computers within the same room or within walking/running distance).
- The Post Office.
- Overnight couriers.
- Modems.
- Ethernet.
- Internet.

Backing up and restoring

How does one back up (and restore, if necessary) important material across several continents?

- Physical magnetic media.
- Electronic media via Internet protocols (email attachments, FTP, HTTP, etc.)
- Downloads and uploads between hp3000 hosts and my Mac PowerBook.

International cooperation

How does one cooperate (and seamlessly merge results) with people who use different languages (human languages as well as computer languages) and software-development environments?

- Physical papers and magnetic media.
- Electronic media via Internet protocols (email attachments, FTP, HTTP, etc.)
- Telephone conversations.
- Face-to-face meetings.
- Video conferences.

Solutions for business transformation

The subtitle of the conference is "Solutions for business transformation". This suits me very nicely, because my whole life has been a series of transformations and my line of work deals with the transformation of database structures.

It is remarkable that, even though every company and institution has undergone unbelievable transformations within the last Internet-oriented decade, the bedrock upon which every business rests has remained quite stable. What is this foundation? Given my background, you will not be surprised by my answer: A solid database management system.

Being solid, of course, does not imply being inflexible. During my presentation, we will discuss the trade-offs involved in sculpting a database for access via:

- Mass-oriented batch processes (which require the best possible throughput).
- On-demand and on-line processes (which require the shortest possible response time).
- Web-oriented browser access (which require the strictest kinds of security and privacy considerations).

Pigeonholing

There are several conference tracks into which this presentation could fit. I took the liberty of including some quotations from Interex's program as I prepared these notes on the conference's subthemes:

- ***E-Services/Internet.*** Gypsy programmers need good computer-assisted work environments and good communications to connect with home base.

- ***General Sessions.*** Gypsy programmers must be generalists and must adapt to different cultures so that we can then practice our specialty with minimal friction.

- ***HP-UX.*** Gypsy programmers must take advantage of technologies that originated with Unix and that have percolated to all TCP/IP-savvy platforms; in addition, you can't beat the schooling — and the examples of what to do and what not to do in life — that you get by "managing, administering, and developing for environments primarily based on HP-UX".

- ***High Availability.*** Gypsy programmers can't afford to be down and must take advantage of "technologies, processes, and support programs that can be used to increase the availability of our data and computing environment".

- ***Linux.*** Gypsy programmers run into more and more of these boxes nowadays — and into more and more Linux enthusiasts on airplanes and technical conferences, not to mention me-too corporate managers.

- ***MPE.*** This is my specialty and MPE business servers are the ones that I use most often for my work. In fact, I have a few hp3000 computers that I maintain on several continents — and carry around in hard-shell suitcases. I use my Mac PowerBook as the hp3000 console (via a serial connection) and as my working ses-

sions (via Ethernet connections). My Gypsy hp3000 computers have survived airline baggage handlers, trains, buses, taxis, hotel bell-people, and all kinds of situations that would have crippled weaker machines. During my presentation, I will show a small collection of photographs that show my Gypsy setups in various continents, hotels and vehicles (including airplanes, boats and trains).

- ***Management.*** Gypsy programmers without good business and technical management skills will have a guaranteed rough time on the road. By definition, gypsy programmers "in leadership roles face many management challenges".

- ***Mobile and Wireless Computing.*** Gypsy programmers are, by definition, highly mobile and must use all kinds of road-warrior computing tools.

- ***Storage Management.*** Gypsy programmers must store — and retrieve — their material with minimal fuss, using sophisticated hierarchies of storage modules, whether in their briefcases or on the other side of the world. Issues of "performance, cost management, data security and future directions" are paramount.

- ***Windows NT and 2000.*** Nobody, but nobody, can get away from these beloved — albeit much maligned — things; even my trusty Mac PowerBook runs Windows 2000 via a software emulator.

Let's discuss the "computer programming" part before the "Gypsy" part

Computer programming is based on simple ideas related to fundamental concepts that are essential to getting anything done:

- Procedure followers (computers and their processors).
- Procedures (computer programs, including operating systems, applications, and tools).
- Data structures (memory and its hierarchy, ranging from fast processor registers to slow magnetic peripherals, as well as the data management systems that keep track of the whereabouts of your information).
- Communications (internal, within the computer and its peripherals, as well as external, with networks of all types).
- Other...

The last few Internet-oriented years have seen a tremendous growth in all of these areas. We will examine examples of each.

Even though the concepts are simple and elegant, computer programming seems to be an impenetrable jungle full of primeval fear and magic.

Most mortals have given up and believe that computer programmers have the power to create ritual recitations of verbal charms or spells ("procedures") to produce magical effects (via the appropriate "procedure followers").

Moreover, computer programmers keep their material hidden away and coded in cryptic languages that make communication with normal people quite impossible.

In a way, this "magic" feeling is justified. Even after decades of programming computers of all sorts in all kinds of programming languages, I am still amazed when a compiler produces object code that a linker consolidates into applications (or applets or libraries) that actually run and "charm" the computer into doing something.

Even the most jaded among you will secretly admit that this is a glorious thing that happens trillions of times a second all over the world (and in outer space), whether you are dealing with super-optimized code that runs in an interplanetary probe at the edge of the solar System or with scripts (the JCL of mainframes or the AppleScript of Macs) that get interpreted by slower virtual machines to manage batch-oriented processes.

Surprisingly, once you are a programmer you are always a programmer and you can program anything. And I mean "anything" quite seriously.

For example, how many of you think of PowerPoint as a venue for expressing your programming skills? And, to boot, I mean your *mathematical and scientific* programming skills! If we have time, I would like to share with you (later on) a little PowerPoint "program" that I showed in a previous talk and that deals with imaginary numbers and the *n* different complex *nth* roots of 1.

Procedure followers (computers)

When thinking about "computer programming", most people think of common computers (mainframes, minis, PCs, Macs) as the procedure followers. There are actually all kinds of procedure followers and they pop up in unexpected places. Here are some examples:

- Personal digital assistants (PDAs).
- Embedded processors (such as the ones inside your car, your cell phone, your digital camera, etc.)
- Java Virtual Machines.
- Web servers.
- Email servers.
- FTP servers.
- Presentation software (such as PowerPoint).
- Web browsers.

In general, anything that you can program, including your VCR's remote control and your car's cruise control, is a procedure follower.

During my presentation, we will have an interactive break where everyone will have the opportunity to identify procedure followers that may not be obvious.

A Gypsy programmer is probably a leading-edge user of many of these gadgets.

Procedures (computer programs)

When thinking about "computer programs", most people think of common applications (payroll, spreadsheets, manufacturing). There are actually all kinds of procedures and they pop up in unexpected places. Here are some examples:

- Perl scripts.
- Stream files.
- Job-control scripts.
- VisualBasic scripts.
- Speed-dial and address-book specifications in your cell phone.
- Source files (which are procedures that compilers follow to produce object files).
- Executable object files (which are procedures that virtual machines or real machines follow to orchestrate computer behavior).

Data structures (memory and its hierarchy)

From single-bit flags to multi-terabyte databases.

- Digital watches.
- Cell phones.
- Pagers.
- PDAs.
- Laptop computers.
- Desktop computers.
- Servers (from small ones for a few users to large ones for thousands of users).
- High-speed (and very expensive) registers.
- "Main" memory.
- Slower (and less expensive) discs (or disks, if you prefer).
- Other kinds of storage peripherals.

- RS-232 (serial).
- Other.
- TCP/IP.
- Various protocols (HTTP, FTP, etc.)

The programming cycle

Regardless of the specific *procedures* and *procedure followers* involved, the fundamental part of computer programming is the ***generation*** of an idea that can be specified by means of some programming language (and implemented by means of some machine that will obey instructions in such language).

Some people believe that the ***writing*** of the idea (in the lingo of some programming language) is the first step in programming.

I respectfully disagree. In fact, I spend a lot more time just staring at the sky than typing on my computer keyboard.

When I find myself staring at the ceiling or at the walls, I know it is time to go outside (under any and all weather conditions) to get some fresh air and some fresh ideas.

This is why I love to go to the Alps, for instance. And this is why I live in Sun Valley, at 6,000 feet (about 2,000 meters) above sea level. My home town of Antigua Guatemala, after all, sits at 5,000 feet (about 1,5000 meters) above sea level, among 12,000-foot volcanos.

Naturally, if you live at sea level (or anywhere in between), you can get your inspiration in ways that suit your style.

Regardless of what you do for kicks, though, you have to come up with ideas before you sit down to write the program(s) that will implement your brain children so that the computer(s) in question will do what you want.

This, of course, is easier said than done. The computer will ***not*** do what you want! Therefore, you have to ***debug*** your implementation.

During my presentation, we will discuss some specific examples of editing, compiling, linking, testing, releasing, supporting, etc.

"Program" vs. "Product"

There is a huge difference between a "program" and a "product" and we will see that it is not easy to go from the former to the latter.

The "Gypsy" part of programming

Back in November of 1978 (almost 23 years ago, as seen from the 2001 HP World Conference in Chicago (Illinois, U.S.A.), I flew from Guatemala to Denver (Colorado, U.S.A.) to give a presentation on the research work that I had been doing with my students at Universidad Francisco Marroquín.

I met Bob Green of Robelle, who invited me to go to Vancouver (British Columbia, Canada). You can see the paper I presented in Denver: *www.adager.com/TechnicalPapersHTML/BreakingFree.html*

Without having ever planned anything, I began my Gypsy programming career right then and there.

Editing

Bob had developed Qedit for his own use as a programmer and had started a small but growing clientele.

I was immediately convinced that Qedit was much better than the Editor that came with MPE but there were some problems with how to run Qedit on the various machines that I was using during my travels.

Please keep in mind that this was 1978, 23 years ago, and there were no such beasts as "personal computers". Only big business had computers (of any kind) and I, personally, did not have any computer at all.

So, Bob was kind enough to develop a "Gypsy" version of Qedit that was licensed to me and not tied to any specific machine. In addition, it had its own versions of the compilers (within the Robelle account) so that I would not be forced to change the SYS account of the people who would be kind enough to allow me to use their computers.

(Qedit uses its own highly-optimized file format which normal compilers don't understand. Therefore, you need to customize the compilers so that they will process Qedit files.)

Backups as dual-purpose mechanisms

My work would live on backup tapes that I would transport between one machine and the next. It was an amazing feeling (and it still is) to arrive at a new place, to restore my files, and to keep on working as if I had always been there, with my environment perfectly set up (thanks to the QeditMgr file that controls Qedit's preferences).

My favorite Qedit configuration is full-screen visual mode. I have heard that I push the limits of the TCP/IP connection between my PowerBook and the hp3000 by specifying 93 (yes, ninety three) rows per screen. I like to have a 21-inch monitor attached to the video port of my PowerBook so that I get two screens and lots of real estate.

It's not unusual for me to have 6 or so screens on the big monitor and a couple of screens on the built-in monitor (including one screen which is connected to the hp3000 through the serial port, as the console).

Later on, I began to use my Mac PowerBook to edit files during trips. So, I developed macros that download all of my source files (in plain-text ASCII format, not in Qedit format) from my hp3000 to my PowerBook.

After having edited the files locally (using BBEdit or the CodeWarrior development environment), I use other scripts that I developed to upload the modified files to the hp3000, using Reflection, which automatically creates the files in Qedit format.

During my presentation, I will share some of the methods that I have developed to move files back and forth among the various computing platforms that I use (and to guarantee a monotonically increasing chronological sequence that ensures that I don't lose work or revert to obsoleted versions).

I have learned that strict discipline and stringent scheduling are necessary to maintain this complex operation in synchrony. As a by-product, I end up with backups in different places (and in different formats). Redundancy is good, but you have to manage it carefully.

Doing many of these tasks by hand would be mind-boggling. So, I rely on scripts that I have created through the years. These scripts are my electronic assistants and they keep track of everything that could go wrong and notify me whenever they detect something untoward.

Single source

I like to have a single "source" for all documents, from which I can produce any kind of output (executable object code, printed documentation, web publishing, etc.)

The ultimate goal is to reduce the number of magnetic media and printed documents and to make everything available electronically.

For instance, I have a lot of material on my PowerBook (including the whole Encyclopaedia Britannica in HTML and all manuals for the hp3000 in PDF).

Likewise, I am working hard to make all distribution of my software via Adager's web and FTP servers.

I will be pleased to share several portable treasures with you and to show as many examples of "moving electrons" instead of "chunks of physical stuff" as time will allow.

A worldwide Chinese puzzle

One day, when I was quietly minding my own business, I saw an email message from the Marketing Manager of HP Asia -Pacific:

> *I would like to know what would be the best way to accomplish data conversion of all user data in an IMAGE database. Our problem is that our customers have data in IMAGE databases in the older CCDC Chinese character set and would like to convert this to the newer Big5 Chinese character set (which is supported by various Windows front ends). They need some utility/tool that would do this and this is where I am looking for your suggestions. At present we have the conversion utility that reads in data from flat files containing CCDC characters and outputs corresponding flat file of Big5 characters. This could be tailored to read/write from buffers.*

Processing "standard U.S. characters" is easy. Dealing with "extended" European characters is a bit more difficult. Actually converting Chinese characters from one representation to another was, up to then, something that I *knew* was possible but had never *actually experienced.*

It is a profound thing, because there is a big difference between *knowing* and *doing!*

The pieces of the puzzle

What a great opportunity for everybody, I thought.

Within Adager, I had the necessary parts for solving the puzzle (rehashing masters whose search fields were affected, rebuilding their B-Tree indexes in the process, rebuilding sorted paths — according to the Big5 Chinese collating sequence — whose sort fields were affected, and so on). But I did not have any idea, whatsoever, regarding Chinese characters (whether in CCDC or Big5 representations, much less the method for translating from one to the other).

Fortunately, HP had some Chinese-conversion software that we could use for the project. I asked René Woc (of Adager) and Stan Sieler (of Allegro) whether they would help me to incorporate this specialized software into Adager's structure. They accepted the challenge with as much enthusiasm as I did.

The players

René spent many long nights and weekends assisting me on the complex tasks and communications involved in this challenge. He remained as "Houston Control" while I was "orbiting" all over the place due to previous commitments that I had already accepted and could not postpone.

Stan Sieler was very helpful with the incorporation of the esoteric translation routines that HP-Taiwan provided. We still have no clue regarding the inner beauty of the Chinese language, so we did everything "by the numbers" and we crossed our fingers, hoping that the mathematics of the whole thing would work out. This was akin to landing a Boeing 747 in heavy fog, exclusively by instruments that you must trust 100%.

It turned out that the original Chinese programmers had already left HP (or had been reassigned) and nobody had any idea regarding the inner workings of these routines.

As a result, I spent a couple of trans-Atlantic flights working on them (using my Mac PowerBook with CodeWarrior's development environment).

After long discussions, Stan and I "massaged" this code and convinced it to work under Adager's hp3000 environment (as dynamically loaded XL library modules).

Meanwhile, HP's MPE/iX engineers were working on leveraging off the work on Chinese Big5 Native-Language Support (NLS) that their HP-UX colleagues had done.

This work was essential for the rebuilding of all sorted paths whose sort fields were involved in the conversion, because the CCDC representation of Chinese characters does not have the same collating sequence that the Big5 representation uses.

Eventually, all of these results converged from all over the world and I was able to orchestrate everything on my traveling European hp3000.

Coincidentally, I did the final work just a couple of blocks away from Vivaldi's church in Venezia. I am sorry I didn't take a picture of my hp3000 (and my 21-inch flat-screen monitor) arriving via a Gondola (I was too concerned about making sure that the equipment would not end up at the bottom of the canal).

But I took these other pictures of that wonderful city so that you may remember this project against the memories of a Guatemalan Gypsy in Venezia.

This example illustrates a major software project that was done in Sun Valley, in California, in Europe, on many airplanes, and via lots of Internet traffic over several continents.

It is very interesting to note that the number of physical packages that were exchanged among the participants was ***zero***. I would say, without hesitation, that our effective use of the Internet was a major factor in our timely success.

I would like to express my public gratitude to Ashish Phillips (of HP in India), to Linda Mei (of HP in Taiwan), and to Tien-You Chen (of HP in Cupertino) for their technical and managerial assistance.

Stan Sieler (of Allegro in California) and René Woc (of Adager in Guatemala and Sun Valley) were integral parts of the lean-and-mean Adager team that supported me as I practiced all the Gypsy theory that I am preaching here.

Our combined work demonstrates that it is possible to deploy a significant software undertaking across the globe, with one programmer moving all over the world using the Gypsy methodologies discussed in this presentation.

Our work also illustrates the value of cooperating with Hewlett-Packard to increase overall good will with users and independent software suppliers.

We managed to engage as many members of the hp3000 community as possible, with different specialties and skills, from all corners of the earth, without going overboard (because it was imperative to maintain a very small and effective group).

In the end, this is a vital issue with Gypsy programming: You can only do the things we mentioned if you are an individual or a small cohesive team. I would not recommend this kind of approach for a bureaucratic bunch.

So, what is a Gypsy programmer?

A Gypsy programmer is not limited to one location, to one computer, to one development environment, to one culture, to one language, to one time zone, and so on.

Would I recommend Gypsy programming for everyone? Obviously not! You must have Gypsy blood as a pre-requisite :-)

Biographical Sketch

F. Alfredo Rego is the Manager of Research and Development at Adager Corporation. He began his career as a university professor (mathematics, physics, computer science) in his native Guatemala. He has given technical presentations on IMAGE/SQL databases throughout the world. In 1981, Interex (The International Association of Hewlett-Packard Computing Professionals) gave him its *Hall of Fame* Award. In 1999, Hewlett-Packard honored him with *The hp3000 Contributors Award* (which recognizes outstanding members of the hp3000 community who are not HP employees).