European Air Traffic Flow Management:
Porting a Large Application to GNU/Linux

Gaetan Allaert
Thales IS sa/nv

Dirk Craeynest
Aubay Belgium

Philippe Waroquiers
Eurocontrol
Contents

• EUROCONTROL/CFMU
• ETFMS architecture
• Application software structure
• Exploratory port
• Port approach and activities
• Portability problems
• Performance on HP and Linux
• Further work and conclusions
EUROCONTROL/CFMU

- CFMU: 2 main roles and systems:
  - Flight Plan Processing: IFPS system
  - Air Traffic Flow Management: ETFMS system

- Operational since 1995

- Development:
  - Using HP-UX, Ada, Oracle, korn shell, Motif, GtkAda, C, UNAS, TCP/IP, ...
  - Under constant functional and technical evolution
  - Migration in 2000
    - from Ada 83 to Ada 95
    - Alsys to GNAT compiler
ETFMS Architecture

- Distributed multi-process application
- Core processes on multi-CPU HP-UX servers
- Client HMI's on HP-UX workstations
- Communication using TCP/IP+UNAS, shared memory
- Duplicated hardware
  - LANs: dynamic rerouting
  - Disks: EMC² raid disks
  - Servers: HP-UX cluster for application switchover

- Aeronautical environment and flight data:
  - Cached in memory for performance reasons
  - Oracle mainly used for restart
Application software structure

- SLOC by programming languages
  - Ada : 1180K 91%
  - Ksh : 78K 6% (building, supervision, ...)
  - C : 45K 3% (Motif binding, ...)

- Sources splitted in subsystems
  - Varying between 10K and 200K SLOC
  - Higher level subsystems
    - provide end-user functionality
    - implemented on top of lower level subsystems
  - Tested automatically by:
    - subsystem specific tests
    - full system tests
Exploratory port

- Evaluate cost/benefit of switching to a new platform
  - Cheaper/more powerful PC hardware
  - Extend the usage of Open Source technologies

- Investigate porting difficulties

- Effort: budget of max 10 person weeks
Port: approach and activities

- Installation and configuration of:
  GNU/Linux, GNAT, Oracle, ...

- Development script porting:
  - Only build and test tools were ported
  - Not included: packaging and deployment, source modifications, ...

- Application code porting:
  - Build each subsystem
  - Validate using the automatic tests
  - Fix non-portable aspects
Korn Shell Script Portability

• 2 categories of problems:
  • related to Korn shell syntax and semantics
  • commands started by the shell

• Many problems:
  • weak syntax checking on HP
  • pipe | execution differs
  • set-user-id bit accepted for scripts only on HP
  • missing commands and/or behaving differently
  • …
C Code Portability (1)

- Many hidden bugs in C code due to:
  - few compile and run time checks
  - weak language definition

- Many problems:
  - `strlen` or `strcpy` with NULL arg is “ok” on HP
  - missing `\0` in string, working by chance
  - header files: addition, removal or re-order were needed
  - array index out of range
  - …
C Code Portability (2)

• Problems seen when C called by Ada:
  – differences in C constants
  – differences in C struct definition 
    e.g. regexp, tm time structure
  – behaviour differences
    e.g. regexp, socket library, ...
  – missing \0

• To isolate Ada from C non portability:
  – generate Ada specs for C constants
  – have more standardized thick bindings
Ada code

• The only problems in more than 1 million Ada SLOC:
  - 2 representation clauses and a bit-mask layout had to be changed due to byte order difference
  - minor differences in a few tests due to the different floating point accuracy in intermediate results
Portability: our findings

- **Shell Scripts:**
  - weak portability, many problems

- **C code:**
  - weak portability due to low level definition and few compilation/run-time checks
  - weak C standard triggers portability problems, encountered e.g. when we called C from Ada.

- **Ada code:**
  - very few problems, related to differences between HP-PA and Intel processor architecture

- **Nr of porting problems (order of magnitude):**
  - Shell: 1 / 100 SLOC
  - C: 1 / 1_000 SLOC
  - Ada: 1 / 100_000 SLOC
Performance: our findings

• Measurements
  • compilation and execution times
  • on 3 machines (HP workstation 400MHz, HP server 875MHz, Linux PC 2GHz)

• Main Conclusion (valid for single CPU):

  1 MHz HP-UX/PA-RISC
  =
  1 MHz Linux/Intel

=> one process runs faster on a 2GHz PC than on an 875 MHz HP server
Further Work

• Port the full development environment
• Packaging and deployment
• Mixed HP/Intel configuration support (HP servers and Linux workstations)
• Supervision
• Support tools: application switchover, backup, computer capacity planning, …
Some Conclusions ...

- The language influences heavily the portability: Ada is a lot more portable

- Avoid calling C code directly => minimize problems by using “thick” bindings

- COTS may help but can introduce potential portability problems

- GNU/Linux is a high quality environment e.g. due to its open nature

- Port goal = study
  Result     = a running system