THE DEVELOPMENT OF SOFTWARE IN THE ADA LANGUAGE FOR A MID-RANGE HEMATOLOGY ANALYZER.

Authors: Robert C. Leif, Ph.D., Jason Sara, Ph.D., Ian Burgess, Michael Kelly, Suzanne B. Leif, and Theresa Daly, Coulter Corporation.

1. Introduction:

1.1. Medical Software Regulatory Authorities:

The FDA is responsible for assuring the safety and effectiveness of medical devices under the 1976 Medical Device Amendments to the Federal Food, Drug, and Cosmetic act. The FDA defines a medical device as:

> "... an instrument, apparatus, implement, machine, contrivance, implant, in vitro reagent, or other similar or related article, including any component, part, or accessory, which...(2) intended for use in the diagnosis of disease or other conditions, or in the cure, mitigation, treatment, or prevention of disease, in man or other animals... (3) intended to affect the structure or any function of the body of man or other animals."

Both the FDA and the ISO 9000 requirements, which will soon be in effect for the European Economic Community, require medical devices including software to be produced according to good manufacturing practices. These practices (Leif and Leif 1992) start with the design input. Both agencies expect to see a formal, documented software life cycle with emphasis on specifying requirements. The FDA expects companies producing software for medical devices to have in house standards for programming. These standards should specify the requirements and restrictions which are to be followed when writing computer programs. The standards should cover documentation procedures, coding standards, and software testing standards. The software should be written in modules which can be tested. The testing process must be comprehensive and well documented. The hazard analysis must be thorough and should be reiterated for each development stage. Wherever possible the system should include fail safes. The code should be inspected and reviewed to ensure that it reflects the design and follows the company policies.

All medical devices which require a premarket notification 510(k) must submit a certification at the time of submission which is a:

> "Written affirmation stating that software development was followed, that good quality assurance procedures were adhered to, and that test results demonstrate that the system specifications and functional requirements were met (FDA 1991)."

1.2. Feasibility Studies: Leif et al. (1991) demonstrated the feasibility of programming a medical device in Ada by prototyping selected components. These components included flow cytometry data analysis software and device drivers for an analog-to-digital convertor and a stepper motor. These authors argued that Ada was superior both in capabilities and safety for programming medical instruments to the then dominant language C.

2. Description of the Clinical Laboratory Instrument:

2.1. Function: The Coulter ONYX, a midrange, quantitative, automated hematology analyzer for in vitro diagnostic use in clinical laboratories, includes a 486 based PC. The purpose of the ONYX is to separate the normal patient, with all normal system-generated parameters, from the patient who needs additional studies. The following hematologic parameters are determined: White Blood Cell or leukocyte count, Red Blood Cell or erythrocyte count, Hemoglobin concentration, Hematocrit (relative volume of erythrocytes), Mean Corpuscular (erythrocyte) Volume, Mean Corpuscular (erythrocyte) hemoglobin, Mean Corpuscular (erythrocyte) Hemoglobin Concentration, Red Cell (erythrocyte volume) Distribution Width, Platelet or throm-

bocyte count, Mean Platelet (thrombocyte) Volume, Lymphocyte percent, Lymphocyte number, Mononuclear cell percent, Mononuclear cell number, Granulocyte percent, Granulocyte number, Platelet Distribution Width, Plateletcrit. Figure 1 is an example of the screen output produced by the ONYX.

2.2. Description: The instrument consists of a diluter, a sample handler, and a Data Management System. The diluter is the primary operating unit of the system. It performs the mixing, transporting, pipetting, diluting, lysing and sensing functions. The sample handler includes: an automated, closed-vial mode that uses a single- tube cap-piercer; a manual, open-vial mode that uses a self-cleaning aspirate probe; bubble/blood detectors; and an optional bar-code reader.

The ONYX employs the Coulter method to accurately count and size cells by detecting and measuring changes in electrical resistance when a particle (such as a cell) in a conductive liquid passes through a small aperture. Each cell suspended in a conductive liquid (diluent) acts as an insulator. As each cell goes through the aperture, it momentarily increases the resistance of the electrical path between the submerged electrodes on either side of the aperture. This causes a measurable electrical pulse. For counting, the vacuum used to pull the diluted suspension of cells through the aperture must be at a regulated volume.The number of pulses indicates the number of particles; the size of the electrical pulse is proportional to the cell volume (Eckhoff 1967, Grover 1972, Waterman et al. 1975, Katchel and Ruhenstroth-Bauer 1976).

The erythrocytes and platelets after suitable dilution are counted and sized together. Since the ratio of erythrocytes to leukocytes is of the order of 600 to 1, the erythrocytes have to be lysed prior to counting and analyzing the leukocytes. Under controlled lysis, a chemical reaction demonstrates three distinct populations of leukocytes: lymphocytes, mononuclear cells, and granulocytes (Cox et al. 1984). Absolute numbers of leukocytes in each category are computed from the WBC histogram. The throughput is 40 or 60 samples per hour, depending on the instrument configuration.

2.3. Data Management System (DMS): This system includes: a computer, monitor, and keyboard. It controls instrument operation, such as the timing and sequencing of the operating cycles. As it receives pulses and raw data from the diluter, it counts, measures, and computes parameters.

The DMS displays, stores and recalls sample data, and lets the operator perform quality control and calibration procedures. It stores patient and quality control data onto its hard disk. With options, it can transmit data to a host computer and archive patient data to disk. The ONYX is shown in Figure 2.

There are several software options. The Extended Patient Data Storage increases the maximum number of patient data records stored in the DMS from 500 to 5,000. Data Archiving allows transfer of patient data files from the DMS data base to diskettes for long term storage. Host Transmission allows the DMS to transmit sample results to a host computer. The Custom Report Designer permits the laboratory to customize the layout of the patient report form. Because these analyzers are exported, the screens must be in the customer's language and the data has to be reported in a format acceptable to the health authorities of the recipient country.

The Quality Control (QC) techniques for the ONYX include: daily instrument checks, commercial controls, X_B Analysis, patient sample review, and interlaboratory comparison.

2.4. Software: The multitude of software functions described above, for the control and operation of the ONYX required 120,000 lines of Ada code. This required 3,470 Person Days, which is equal to 34.6 lines of code per day. The Alsys 386 DOS compiler took 8 hours on a 25 MHz 486 to compile this code with optimization. Most of the software for the instrument was developed in Alsys 386 Ada compiler supplemented with Aetech Ada_User libraries. A secondary microprocessor which performed communication functions was programmed in C because it permitted reuse of existing C code and Alsys Ada did not support existing Intel in circuit emulators.

3. Software Requirements for a Commercial Ada Application:

3.1. Translatable User Interface: Commercial applications such as Coulter's biomedical products are sold to a diverse set of customers. The user interface must be translatable into multiple languages without modifying the code. This has proven to be a problem with Ada because of the limited character set defined and lack of third party applications.

3.2. Attractive User Interface: Unlike in a large government contract, customer purchase decisions are made after the product is fully developed. User interface concerns are therefore more important to project success.

3.3. Cost Sensitive: Originally, an 80386SX industry standard personal computer was selected. This was later upgraded to an 80486SX. These computers including peripherals are much less expensive than a comparable custom embedded system or a workstation. One reason for choosing Ada was the availability of 32-bit support software on a low cost target platform.

3.4. IBM^R PCDOS: This operating system is ubiquitous, inexpensive, and provides a pseudo-embedded system. Abundant off-the-shelf software including device drivers is

available. DOS provides minimal interference with Ada tasking. For true real-time performance, programs need to employ Ada tasking to directly control the disk. A subset of the POSIX specifications with DOS bodies should be developed. The same should apply to the Macintosh.

4. Beneficial Features of Ada:

4.1. Standardization and Validation: The Ada language is described by ANSI/MIL-STD-1815A, accepted by ISO, and described in a U.S. Government official reference manual. Validation & WARRANTY. Only Ada compilers pass a Unites States government approved validation suite.

> "Alsys warrants that the Compiler Programs and/ or Executive Programs perform substantially as described in the Documentation and the Reference Manual for the Ada Programming Language."

4.2. Built in Modern Software Engineering Features: Ada is a portable, general purpose language. Since it is based on Pascal, it is readable and self-documenting. Safety features include: strong typing, range checking, and exceptions. Exceptions map to a subset of hazards. The package structure provides the modularity required for a group to develop code. Real-time constructs based on tasking are included in the language.

4.3. Established Methodology and Software Engineering Culture: The requirements of the Defense Department have resulted in the establishment of independent verification and validation organizations. Well engineered code is often available from software repositories. Since Ada is a product of the software engineering establishment, its practitioners often see the attempt to achieve zero defects as a moral booster and a professional obligation.

4.4. Project Management: The Ada package structure permitted partitioning of the software into manageable components. These packages were easily traceable to the software requirements and were the basic structuring mechanism for the software system. The software project schedule throughout the development life-cycle was monitored in terms of the Ada packages. The separate definition of interfaces (specifications) and their implementations (bodies) in Ada allowed an orderly integration of the software components and reduced the integration time.

4.5. Comparison with the Nearest Competitor, C++: At the time the choice of language was made, no 32 bit C++ compiler for DOS was available. Portability of C++ code has yet to be demonstrated. The presently published data indicates that Ada is still safer (Tang 1992) and more cost effective than any third generation language now known (Mosemann 1993). Most of the deficiencies of Ada have been remedied in Ada 9X (Taft et al. 1992) and 9X has a better design for object oriented programming than C++. C++ compilers in general have more user friendly front ends and easier interfaces to most commercial off-the-shelf software. The most significant advantage of C++ is that it is being very ably mass marketed by major software corporations.

5. Ada Problems:

Many of the problems which were experienced with Ada 83 are solved in Ada 9X. These problems with Ada 83 were observed by programmers who were creating software in the medical instrument industry. Since they were not involved in monitoring or generating the Ada 9X requirements, their comments add significant credence to the necessity for the changes incorporated in 9X.

5.1. Foreign Language Problems: Ada 83 does not support double byte characters for use in implementing Japanese. An international character set is not supported and characters were limited to 128 items rather than the 256 available in most languages. This has recently been remedied by action of ISO WG9 and is fixed in Ada 9X. An Ada 83 solution would be for type CHARACTER to be a GENERIC under Gen_Text_Io. This package should be delivered with an instantiation as Text_Io for ASCII and as Latin_Text_Io for Latin 1. The implementation of a CHARACTER CLASS for Ada 9X would permit the flexibility required to produce screens readable by the laboratory technicians or their equivalents of all nations.

5.2. Object Oriented Constructs: Ada 83 does not support pointers to functions. This has been solved in Ada 9X. The 9X solution of pointers to functions should be implemented for the next releases of Ada 83. This would clean up the X Windows interface and facilitate the development of a Microsoft^{R} Windows interface**.** Ada 83 does not support inheritance. Many believe that this makes it difficult to build reusable code. Generic templates provide a functionality which is different from that provided by C++ classes. The combination of single inheritance and generics specified for Ada 9X provides an elegant, safe solution (Banner and Schonberg 1992).

5.3. User Interface Problem: The Graphical User Interface Designers for use under Alsys Ada for DOS are primitive and do not have adequate features for real-time commercial applications. Extensive in-house resources must be used to implement a GUI. Before the choice of programming language was made, the original screens were prototyped with a tool that supported virtually all of the major programing languages: Pascal, C, dBase, Modula2, etc. It did not support Ada. Significant commercial use of Ada requires that screen development tools become available. AdaSage should be enhanced or other GUIs which have been developed for POSIX be ported to DOS.

5.4. Third Party Library Problem: Third Party

libraries such as math packages, graphics, and databases for Ada are primitive. The few commercial third party applications which support Ada do not support Ada's tasking constructs, as they were originally developed for non-tasking languages such as C. The set of libraries provided by each compiler vendor is very different, making it difficult to port Ada code from platform to platform. The set of libraries is very limited and provides inadequate functionality.

The Annexes for Ada 9X provide a partial solution. Hopefully, there will be a mechanism to create new annexes and other associated secondary standards in a timely manner. The Ada community including the DoD purchasing organization must proselytize the commercial vendors to include in their products Ada interfaces and code generators.

5.5. File I/O Problem: Ada's Standard Packages for File I/O do not allow the mixing of records of differing data types in a single file. Since Direct_IO is a generic, it can only be instantiated with one data type. The Text_IO package does not support appending to a nontext file. These limitations of Ada's File I/O facilities required the use of nonstandard packages. Alsys Ada provides a good package for File I/O under DOS, but the 'Address clause must be used, making it likely that memory will be overwritten. Saving complex structures, especially those with discriminants can generate erroneous code. A higher level I/O package is needed.

A partial solution would be to use the POSIX_IO specification with a DOS body. For instance, there is an Append option for the function POSIX_IO.Open_or_Create.

5.6. Mixed Language Problem: Alsys Ada 386 does not support interfacing to any languages other than C and assembler. Two significant marketing errors were not to include an interface to Intel's PLM compiler and lack of support for Intel in circuit emulators. Callback into Ada is not supported. Ada 83 does not require any language interface capabilities. The design of Ada 9X facilitates interfacing with other languages. For commercial use, an interface to COBOL is urgently required.

Enhanced libraries and tool kits in Ada will reduce the necessity to interface with other languages. An X Windows and a real time POSIX written in Ada would transfer the problem to the C and C++ vendors.

6. Alsys Ada 386 for DOS Advantages:

The Alsys compiler produces true 32 bit code, is not limited by the DOS address space, and can manipulate large arrays. The code is fast and efficient. Alsys provides a complete and production quality implementation of the Ada language, which is capable of handling large numbers of tasks and works with nested generics. The majority of the time it provides excellent error messages; however, sometimes they are useless. Alsys has an extraordinarily powerful library manager. However, the design can be thoroughly intimidating. Lastly, a powerful Ada oriented debugger was provided; however, see below.

7. Problems with Alsys Ada:

7.1. Alsys Debugger Problem: The Alsys Ada debugger for the 386 for Version 5.1 had many serious problems; release 5.1.1 remedied or ameliorated many of them. Since 5.1 was current when the code was written, it was very difficult to debug programs. Examining variables often resulted in the debugger crashing. This was very serious with 5.1 and occasionally occurs with 5.1.1. It might be the result of the lack of protection afforded by the DOS operating system. No information on the state of tasks is available. The Alsys debugger requires excessive memory (about 10-16 megabytes) compared to those for other languages. Our programs often operate in graphics mode. The graphical output cannot be swapped to the screen while operating in the debugger. The human interface lacks features that are commonly available in most debuggers. For instance, the cursor must be placed on a variable to examine it; rather than, entering the name of the variable and employing it as a watch point. It should be possible to automate the present object examine command in conjunction with breakpoints. The debugger is slow. This is very noticeable when stepping through code which has slice arithmetic e.g.: a(1..n):= b(1..n) takes an extremely long time and is dependent on the slice size!

An enhancement which would facilitate the debugger's use is direct execute-to-cursor without having to set and unset breakpoints. Breakpoints disappear when an exception occurs when stepping through the code.

7.2. Run Time Error Problem: Alsys Ada is brilliant at elucidating potential run time errors. This is a very valuable facility since the compiler has raised constraint_errors during the actual testing of the product. Constraint_errors are described as exceptions and the line numbers and subprogram locations are given. However, no means is provided to generate error messages of the quality of those produced by the compiler.

For instance,

"Array25Program terminated Unhandled exception **CONSTRAINT_ERROR** (numeric overflow)

Line Subprogram name Compilation unit name

379 SHIFT_AND_SORT_ARRAY_AC.ARRAY256

282 SUB3D. SUB3D

7.3. Listing Line Number Error Location Problem: The full listing of the source code produced by the compiler is 132 characters wide and thus cannot be printed out with an 80 column printer. Since the statements which give the

line number of the error locations are right justified, they are off the screen.

8. Conclusions:

For a medical device, the benefits of Ada outweigh the disadvantages. The syntax and capabilities of Ada 83 are well suited for the efficient development of the quality expected for a medical device. The enhancements provided by 9X will even further the lead of Ada over competing languages. The problems with Ada are not inherent to the language. Ada, because of the captive DoD market, is technology rather than market driven. Ada vendors have been shielded from the normal market forces which have resulted in excellent user interfaces and tools for other languages.

Presently, the Ada language design mechanism is based on the waterfall model. A mechanism must be developed to permit correction in less than 10 years of minor but painful design mistakes such as the lack of a call back or the limitation of the representation of characters to 7 bits. A constrained spiral model permitting minor downwardly compatible changes and the development of new annexes would provide the flexibility required for Ada to survive and thrive in the competitive commercial world.

9. Acknowledgments:

This project would not have been possible without the excellent instruction in Ada and software engineering provided by Eugene Bingue, Westly Mackey, and Ben Brosgol. The Center for Health Technologies of the State of Florida facilitated this transfer of DoD technology.

10. References:

1. S. B. Leif and R. C. Leif. “Producing Quality Software According to Medical Regulations for Devices”, Computer Based Medical Systems, Proceedings of the Fifth Annual IEEE Symposium 265-272, 1992

2. Office of Device Evaluation Center for Devices and Radiological Health Food and Drug Administration, Department of Health and Human Services, Public Health Service. “Reviewers Guidance for Computer Controlled Medical Devices Undergoing 510(k) Review”, Stamped Aug 29, 1991.

3. R. C. Leif, I. Rosello, D. Simler, G. P. Garcia, and S. B. Leif. “Ada Software for Cytometry”, Analytical and Quantitative Cytology and Histology 13 440-450, 1991.

4. R. F. Eckhoff. “An Experimental Indication of the Volume Proportional Response of the Coulter Counter for Irregularly Shaped Particles”, J. Sci, Inst. 44 648-649, 1967.

5. N. B. Grover, J. Naaman, S. Ben-sasson, and F. Dojanski. “Electrical Sizing of Particles in Suspension III. Rigid Spheroids and Red Blood Cells”, Biophys J,; 12 1099-1116, 1972

6. C. S. Waterman, E. E. Atkinson, B. Wilkins, C. l. Fischer, and S. L Kimsey. “Improved Measurement of Erythrocyte Volume Distribution by Aperture-Counter Signal Analysis, Clin. Chem., 21 1201-1211, 1975.

7. V. Kachel and G. Ruhenstroth-Bauer. “Methodik and Ergebissne Optiseher Formfatorunter-suchungen bei der Zellvolumenmessung nach Coulter. Micros Acta”, 75 419-423 1976.C. Cox et al. “Evaluation of the COULTER_ Three-Part Differential System”, Am. J. Clin. Path., 82 372-373, 1984.

8. L. S. Tang. “A Comparison of Ada and C++”, Conference Proceedings TRI-Ada ‘92, 338-349, 1992.

9. L. K. Mosemann. “New is Good--Is Ada Too Old”, CrossTalk, The Journal of Defense Software Engineering, 40 8, 1993.

10. T. Taft et al. “Ada 9X Mapping Specification 4.6 & Rationale 4.1”, Intermetrics Inc. Cambridge MA, 1992.

11. B. Banner and E. Schonberg, “Assessing Ada 9X OOP: Building a Reusable Components Library”, Conference Proceedings TRI-Ada ‘92, 79-90, 1992.

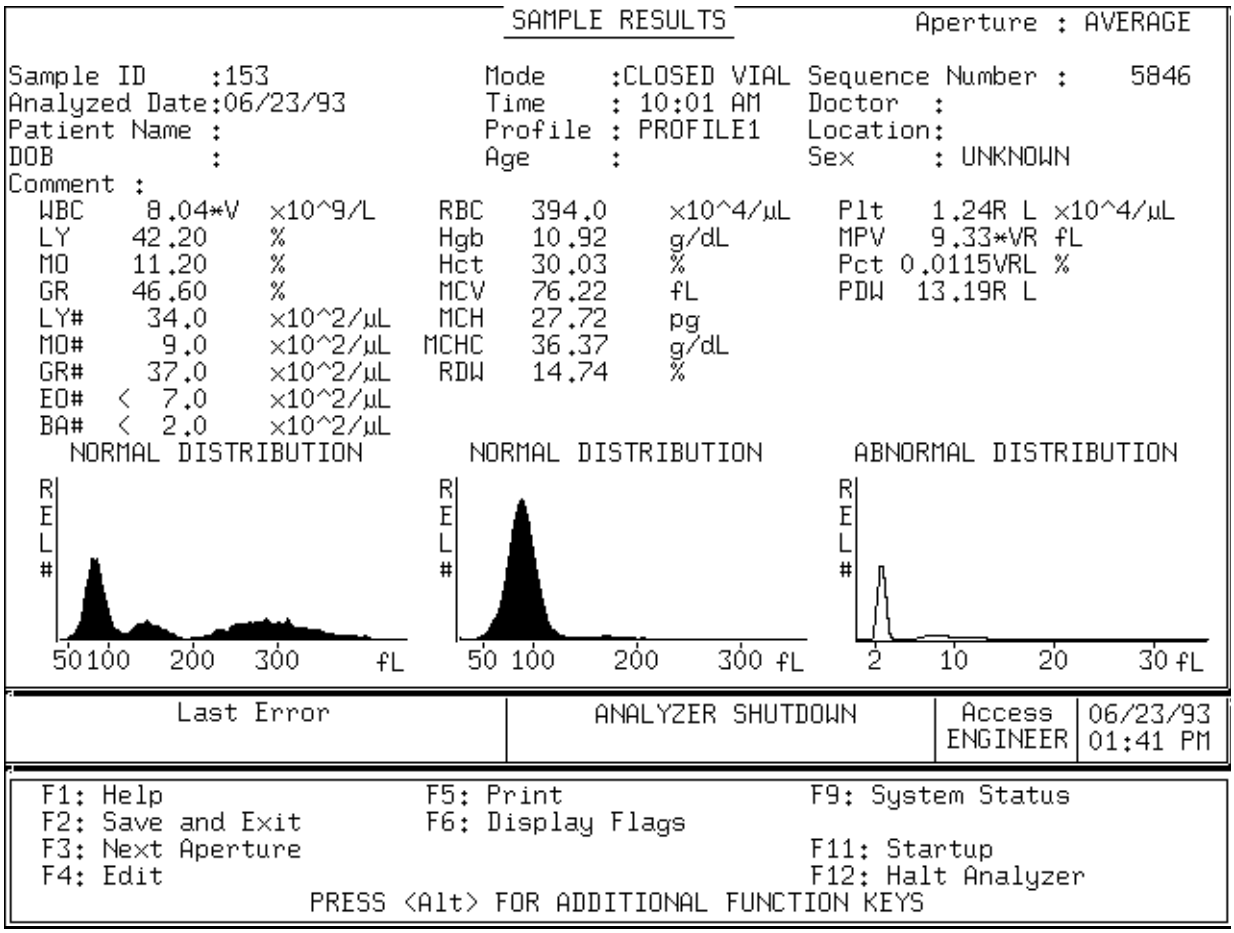

Figure 1. The instrument printout (top) shows the absolute numbers, percentages, and sizes of the blood cells as well as providing hemoglobin data. The printout (bottom) shows the histograms of the white blood cells (WBC), red blood cells (RBC) and platelets (PLT). The ordinate is the relative number of cells. The abscissa is the volume in femtoliters.

Figure 2, Coulter^{R} ONYX