ADA AND OBJECT ORIENTED DESIGN FOR SIMULATION IN THEORY AND IN PRACTICE

P. Baker
Marconi Simulation
A Business Unit of Marconi Instruments Ltd.
Nr. Dunfermline, Scotland

ABSTRACT

For any software development group attempting to establish its future strategy for major software projects, there has never been a greater range of languages and techniques from which to choose. For those with major defense industry commitments there is overwhelming pressure to conform to standards, and in particular to standardize on the use of Ada.

In this paper the particular experience of one group of engineers who followed this path for the development of an avionics sub-system Simulator and Test Rig are discussed. The objectives and key decision points are highlighted and, although the project has not yet reached it's completion, there are significant conclusions that have been identified during the development process.

Consideration is given to:-

Primary Objectives
Clear representation of the Requirement and its reflection in the Design.
High level of software component re-use.
Efficient development process.
Maintainability of the product.

Methodology
The Codes of Practice used in each phase of the software LIFECYCLE.

Development Environment
Ada products used.
Software tools.
Workstations and support equipment.

The Development Process
Reactions from the software team.
Training needs.

The Results
Costs and Benefits.
Achievement of objectives.

In conclusion, this paper summarizes the degree of success encountered with this approach to Ada and reviews plans for 'next time'.

INTRODUCTION

Early in 1987 Marconi Simulation realised that a change from the existing dominant language (PASCAL) to Ada would probably become mandatory within the foreseeable future and a specialist Ada Software Engineering group was established to identify the implications of making this transition. This group focussed on acquiring knowledge of the Ada compiler/APSE marketplace and on the capabilities of the Ada language; scope was also extended to cover a broad scan across the Software Design Tools market and to conduct a program of product evaluations.

The group was expanded in early 1988 and tasked with sharpening up the evaluation process and with identifying a specific development environment, compiler and design philosophy for use with Ada; the projected timescale for achieving this objective was 12 months.

During this period it became clear that there would be increasing pressure both internally and from the UK Ministry of Defence (MoD) for any new work to be bid in Ada: In the event the first project to be bid in Ada was successful and only 6 months into the evaluation program the prototype 'project' being used for technique development was replaced with a 'real project' on which to

cut our teeth. This immediate 'total immersion' approach to a new software development context, although in many ways less than ideal, does concentrate the mind wonderfully and undoubtedly caused a significant need for pragmatic engineering decisions and a critical appraisal of priorities.

The decision had by this time been made by the Ada Software Engineering group that we would use SUN 3/Series workstations as the basis of the Software Development Facility; we were targetting onto 68020 VME systems and this choice offered a degree of compatibility and was well supported by vendors at competitive prices. It was realized that workstations equipped with local hard disks would be essential to provide the necessary level of compiler performance required to achieve a high level of programmer productivity. The planned developments would require the assembly of a large number of workstations on a network and previous experiences had highlighted the likelihood of delays caused by high network traffic when using large numbers of diskless units.

THE PROJECT (NST)

The software development project that forms the basis for this discussion was placed in October 1988 and is for a Software Test Rig to allow testing of major avionics software sub-systems within the Nimrod ASW aircraft. This rig will be used by the Nimrod Software Maintenance Team (NST) for checkout and test of modifications made to the four major sub-systems namely:-

Central Tactical System,

Electronic Surveillance Measures,

Radar,

Acoustics.

The Rig consists of five modules, each with approximately the same general configuration but differing in signal stimulation capabilities.

The basic module consists of a PC(386) based Control Workstation running Microsoft Windows software linked by Ethernet to two VME bus based processing units, namely:-

1. The Master Processing Unit (MPU) which houses the Scenario generator software and supports special interface stimulation hardware and link control units.

2. The Operator's Monitor Panel (OMP) interface. This is a replacement for an existing piece of hardware that allows direct monitoring and control of the CPUs within the relevant aircraft systems.

Control over both the MPU and the OMP is via the controlling PC unit.

All five modules are linked together on a common Ethernet.

The Ada context for this project includes only the 68020 based VME units, ie the MPUs and OMPs. The timescale for the project was nine months for the initial phase (one unit as above for ESM only) with all remaining units following as a phased program during the ensuing 12 months.

The similar nature of the modules points to there being scope for re-use of code (this was identified as an important objective) not only across the various phases of this project but to point the way for other subsequent projects as a major cost reducing factor. Initial work during the bid phase indicated that the whole project, excluding the Man Machine Interface (MMI), would involve approximately 20,000 lines of Ada code.

THE SOFTWARE ENGINEERING TEAM

The software team for the project was assembled from a nucleus of very experienced software engineers with a considerable level of prior knowledge of this type of product. The software manager and software technical

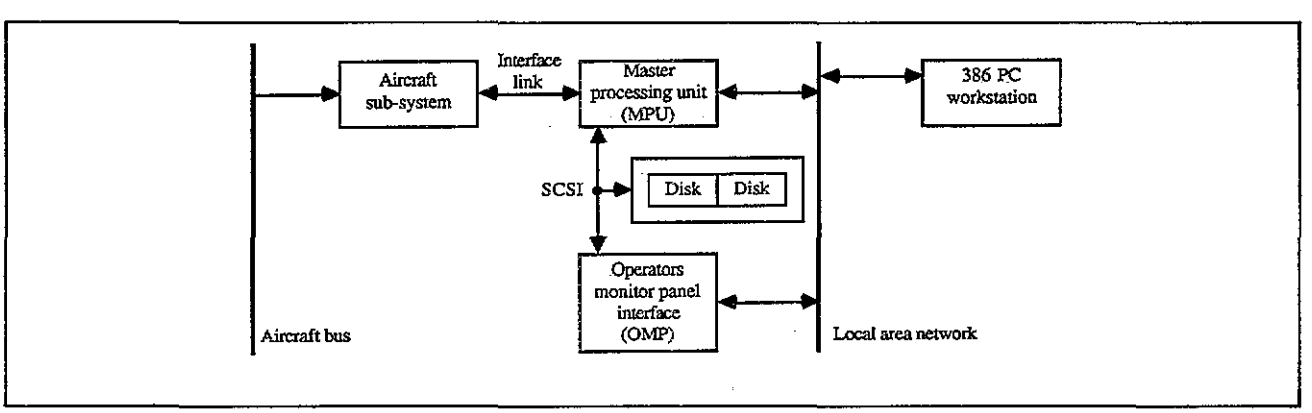

Fig 1 NST Module Configuration

project leader are both very experienced in the arena of Nimrod aircraft sub-systems (5 to 10 years). Marconi Simulation have previously designed and built a number of successful Full Mission simulators for this aircraft, in which the accent was on aircrew training, subtly different from operations as a software test rig. The remainder of the team for the first phase consists of four software engineers with experience ranging from 2 to 4 years in simulation.

Each phase of the project required teams with specialist skills deriving from the individual sensor systems (ESM, CTS, Radar and Acoustics) and all previous software work had been in PASCAL. No software team member had any more than a few weeks prior exposure to Ada although this did include a formal one week Ada Training course.

The project team was reinforced by a five man Ada Software Engineering team (the team running the Ada evaluation program); this team varied in strength and content over a period but generally consisted of software engineers within 5 to 10 years experience, tasked to contribute to the main project in the following key areas:-

1. The establishment of core software to handle the PC-VME Ethernet link.

2. The establishment of basic Disk File I/O facilities and basic system software utilities, *including boot facilities for the target VME sub-systems.*

3. Production of Design procedures/Codes of Practice and coordination of the design methodology across all phases of the project, including a basic level of informal training in design techniques.

4. Support of the ever growing Software Development Facility.

5. As a skills centre for Ada techniques and as a focal point for the control and coordination of Ada specific problems.

6. Coordination and monitoring of the effective use of the Software Design Tools.

7. Updating of working practices in the light of ongoing experience.

8. Provision and enhancement of Software Configuration Control procedures and facilities.

9. Gathering information on actual code generation to help calibrate future code estimating activities.

Both the Project Engineering team and the Ada Software Engineering team operated in close cooperation under the direct control of the project manager.

THE FUNDAMENTALS

During the short evaluation phase that preceded this project a number of key observations had been made by the Ada Software Engineering group. These can be summarized as follows:-

1. Ada on its own would not provide the answer to the software engineer's prayer but it could be an important enabling factor and a necessary first step to improving software quality and engineering productivity.

2. We had previously suffered from a fundamental lack of a 'design process'. Although design was being partitioned in accordance with the traditional software lifecycle there was no specific set of practices to be implemented to help a software engineer develop an 'effective' design.

3. We had acknowledged the traditional lifecycle phases for software but had addressed the Requirements Analysis aspects in an unstructured manner and had no natural flow from Requirements Analysis into the Design Process.

4. Despite 'best intentions', coding was in general being started too early and the formal set of diagrams and detail design documentation were often being generated retrospectively - in fairness, the high level design documentation was generally well prepared but was seldom driven down to a sufficiently detailed level to effectively support the major thrust of the design activity.

5. Design Reviews and their associated Code walkthroughs were always based either on ad-hoc text or diagramatic representations of the design or at a later stage on the code listings; in consequence these reviews were seldom complete or effective. Something better was needed and hopes were high that a Design Tool might meet this need.

6. No Software Tools were available in support of the Requirements Analysis or Design phases and this was a major contributory factor to the perpetuation of old 'bad habits'.

7. To merely train engineers in the syntactical use of Ada would compound an existing problem. (Hacking code in Ada with a large slow and expensive Ada Compiler is a worse sin than hacking in most other languages.)

8. The key was to train engineers to 'design software' in Ada, not simply to write code in Ada.

Having made these observations, there came the following more difficult questions:-

1. Could the necessary decisions be made in time for the NST project?
2. Could Codes of Practice be evolved and disseminated quickly enough to be of use to the project software designers?
3. Would capital costs be prohibitive?
4. How would we train the engineers?
5. Would the theories work in practice?
6. Would rushing this approach and possibly achieving only a limited success cause damage to a worthy longer-term objective?

The balance of probability indicated that there was little merit to avoiding any of these issues and the decision was taken to address as many of them as could reasonably be handled within the project timeframe. Technical cover would be provided by aligning the Ada Software group directly with this project both to minimize risk and to extract the maximum longer term benefit.

Methodology and Tools

The key to the design process for the typical Simulator/Trainer had to involve a mechanism for reliably tracking from Requirements Analysis through to the design, in a manner that retained the shape of the problem in a form understandable to both the customer and the engineering team. Previously all design had been based on the concept of functional decomposition with top-down design and 'information hiding' to help encapsulate components of the design. This approach tended to be born of the design process and not naturally generated from the Requirements Analysis phase. Designs following this approach have no explicit encapsulation of function and data, and the route to achievement of re-useable components is not explicit.

The prime motivation to achieve a design process that encouraged the maximum level of component re-use coupled with the 'real-life object' nature of simulation encouraged the promotion of Object Oriented Design (OOD) as the mechanism for generating the Design process; this in turn indicated that the optimum Requirements Analysis process would similarly be Object Oriented (OORA).

At this point we encountered a basic problem in that there did not appear to be any significant source of supply of CASE tools to support Object Oriented Design, nor indeed any detailed and explicit mechanisms for achieving OORA and OOD.

Initial forays into the Design tool arena highlighted the ready availability of tools to support the functional decomposition/data flow philosophies, viz Yourdon/DeMarco and extensions due to Constantine and Hatley, but little that was designed for OOD.

Evaluations were limited within the timeframe to MASCOT 3, TEAMWORK, Software Tho'Pictures and CARDtools - of these, CARDtools was selected as being the most appropriate for representing OOD although not being specifically designed to do so. On the Requirements Analysis front we did not find a support tool but have established Codes of Practice that owe something to both CORE (Controlled Requirements Expression) and HOOD (Hierarchical Object Oriented Design); the preparation of these Codes of Practice straddled the first phase of the NST project ie the Codes of Practice for Requirements Analysis was not available for the first phase - (ESM), but were available for the subsequent phases.

Compiler

The Ada compiler selection narrowed to a choice of Alsys, Verdix or Telesoft. The Telesoft product was perhaps the logical choice as this was also supported by Ready Systems, the suppliers of the CARDtools, Design tool and performance figures were available for incorporation into this design tool. However, as there appeared to be no significant performance differences between these products and the reviewers overwhelmingly preferred the user interface to the Alsys product, this was adopted. A spin off benefit of this choice was the availability of a PC-hosted, cross-compilation facility and a high level of engineering support in the UK.

It was felt that the specialist run-time kernel offered by Ready Systems, although potentially attractive, did not out-weigh the 'ease of use' and technical support issues.

Other Aspects

Two aspects of the targetting of Ada onto the VME 68020 CPUs posed a problem. The first was the absence of a target system disk file input/output facility from the compiler vendor (an almost universal situation it seems!). Secondly, we required an Ethernet capability and the ability to pick up an 'off the peg' standard system would have been attractive. Both of these issues had implications that could potentially overpower other factors in the choice of a compiler vendor.

Disk File I/O. This complication was resolved relatively quickly by agreement with the compiler vendor that they would accelerate an existing program to achieve this functionality and would work together with Marconi Simulation to produce a fully host/target compatible file I/O facility. For our part we particularly wished to have close liaison with the supplier in this context as we needed to develop the relevant SCSI disk drivers and a simple protocol for achieving multi-processor boot facilities from a shared Winchester disk drive.

Ethernet Connectivity. The initial intention here was to adopt the TCP-IP protocol as standard and to buy-in the necessary software components. This proved to be

problematical since there were complications with using TCP-IP at the PC end and we had specific reasons and strong motivation to base the PC-MMI system on a Microsoft Windows development. We were therefore left with two options:-

1. A home-brew or collaborative TCP-IP venture to produce a Windows compatible facility.

2. A simple, low-level protocol implementation at both VME and PC ends.

We opted for the latter approach on the basis that we felt the need to control and establish an efficient low-level protocol on Ethernet that was appropriate for 'real-time' use, and not risk being in a single source situation with a third-party solution.

THE DEVELOPMENT EXPERIENCE

A synopsis of the views of each of the following major participating groups within the NST project is presented below:-

Groups - *Project Management*
- The Project Software Manager and the Technical Project Leader
- The Software Development Team

Project Management

This has probably been the area that has seen the most 'concern' over moving to a new approach to software engineering, however, faith has been retained throughout. The primary areas of concern expressed during the development have been:-

1. The high cost of populating a multiple workstation networked software development facility for use with Ada.

2. The need to quantify and fund the appropriate number of Design tool equipped workstations with sufficient power to effectively service an Ada-based project development.

3. The incremental costs for additional host and target cross-compiler and development utilities for large software teams.

4. A major training program on three concurrent fronts, namely:-

   - Software design
   - Design tool use
   - Ada language

5. The continual need to assess and resource the high entry costs for this development with no guarantee of future cost savings.

Software Technical Management

At this level those involved were significantly aware of the inadequacies of previous software development strategies that new ideas were greeted with enthusiasm and readily absorbed.

Once the engineering teams were established, the real problems of managing the training, analysis and design using the design techniques and tools rather than rushing directly to Ada coding had to be carefully managed.

The major problem was that Ada acts as a beacon for programmers who can easily be tempted to rush past the more important Object Oriented Analysis and Design activities in their enthusiasm to get into the details of the language. This problem was made more acute by the fact that we really did need to push rapidly through to Ada code in some areas in order to establish some important metrics that were perceived as critical to the overall system design.

Key points have been:-

1. Ada Tasking:- There is a great temptation for programmers to make the maximum use of Tasking and it rapidly became clear that although a high level design may be constructed that uses many tasks, in general the 'real-time' price paid for this can easily be excessive. This proved to be a less serious problem in retrospect because the design can generally be amended at a later stage such that Tasks are grouped and, if necessary, replaced by simple sequentially scheduled packages without compromising the fundamental design concepts.

2. OOD:- The OOD concept and associated Codes of Practice have been extremely well received as very effective mechanisms for 'getting to grips' with the design in both the 'problem' and 'solution' domains. Such simple requests as "Write down what you need the system to do - completely - on one 11" x 8" sheet of paper and identify the nouns as the primary objects" have repeatedly been found to be astonishingly illuminating and helpful.

3. The Design Process:- The OOD concept has proved to be quite an elusive target. The tendency for users to slip back into old ways and decompose 'functions' rather than 'objects' has been noticeable, largely due to the lack of significant literature to cover the OOD process and the relatively late arrival of our own Codes of Practice with their associated worked examples.

The Software Engineers

The OOD concepts were welcomed but not immediately understood and only when worked examples were introduced was this technique thoroughly absorbed. It is generally hard work to get to grips with any new

Design Process, and to struggle with this at the same time as learning a Design tool and the Ada language was found to be a heavy load at the start of a new project.

The development of effective working practices was achieved partially as a result of observing the problems of the engineers in grasping the design task; this resulted in the realization that during the early stages of evolving a design, a design tool is not necesssarily helpful unless its MMI is at least as effective as the good old paper and pencil. The result of pushing engineers too early onto the CARDtools facility proved to be demotivating.

Much of the team's early work was cast aside as they developed the design concept; the tool being less than optimum for this sort of fast iterative process, manual methods proving more satisfactory until the design stabilized. CARDtools came into its own once the design had stabilized to allow the rigorous checking of completeness and standardized design representation that is essential.

SUCCESSES AND FAILURES

Starting with the most successful and working through to the less successful.

Object Oriented Requirements Analysis

Top of the list must be the acknowledgement and formalizing of the Requirements Analysis phase with the introduction of an explicit set of practices to be employed. This ensures that the customer's needs are met and contractual obligations fulfilled in a manner that feeds directly through into the Design Process.

Object Oriented Design

A very close second, without which the above would not have fitted into place so neatly, comes the Design methodology - the adoption of Object Oriented Design practices. This has opened the door to explicit mechanistic ways of thinking within both the 'problem' and 'solution' domains (Requirements Analysis and Design) that yield readily checkable, complete and encapsulated designs. This directly supports one of the major targets, that of encapsulated and consequently re-useable code.

The Design Tool (CARDtools)

This has been successful in so far as it provides our only mechanism for checking the completeness, if not the correctness, of designs and has provided the third most successful item, the standard representation for designs; these hierarchical decompositions into objects (packages) and within packages a hierarchical decomposition by function, although not ideal, have provided the common ground on which design ideas can be reviewed, design assessed and generally has served as a forum for the establishment of ideas about what the ideal design tool could and should offer.

The existing PDL within CARDtools is not AdaPDL, although this is promised and should potentially transform the tool as it will then have the potential for direct Ada code generation. I feel sure that the design tool market is an enormous area of potential which has yet to be successfully addressed for Ada.

CARDtools is the best option we have found so far but there is plenty of scope for development in this area.

The Compiler

The decision to use the Alsys compiler has proved sound - the disk file I/O facilities were integrated without undue complication and are functioning well. Code performance has generally been excellent with only a limited number of minor bugs being unearthed in the compiler. There is still a restriction to use of RS-232 for target code download but a move to Ethernet is anticipated in the future - this has proved a little tedious.

Absence of an APSE

Time pressures prevented any significant evaluation of APSEs other than having very quick look at GEC-Marconi's GENOS; perversely this probably rates as a success in that despite the fact that we are using an ad-hoc collection of loosely coupled tools and utilities, this has allowed us to learn about the pitfalls without being constrained by an APSE environment, many of which are rigid or difficult to tailor.

Development Facility

This has been a successful choice of equipment but the complexities and management requirements of a growing network of SUN workstations were underestimated. It became clear very quickly after taking ownership of the first three workstations that the management and configuration of the equipment to achieve optimized configurations for each software tool being used was a considerable discipline in its own right, and required dedicated staff. We currently have two to three full-time software engineers dedicated to managing and steering the growth of this complex and it seems unlikely that this will decrease with time. This overhead was not fully appreciated or budgetted at the start.

Software Configuration Control

One of our current known omissions has been the achievement of a Software Configuration Control process that is as automated and integrated as we would like. This is probably related to the APSE question since an APSE would almost certainly have carried provisions for Configuration Control.

We are however making effective use of manual methods involving a general purpose database to hold Build Lists, all Software Configuration Lists and references to current and historical development files. The files themselves are held separately in simple password protected directories with access priviledges noted in the database. This system works, given effective policing, but is not appropriate in the longer term and is an area

requiring priority upgrading - hence this must be rated as at least a half failure.

Re-useable Code

The principle of achieving code re-use through OOD and Ada has been proved. The only incomplete success really concerns the extent to which the project code being produced is truly emerging in a re-useable format and in an OOD context; this in truth has proved to be very patchy, the reason for this probably being that Codes of Practice arrived too late in some instances and time pressures did not allow the necessary re-work. This problem was anticipated and is being addressed by specifically reviewing the project code in parallel with the development exercise in an attempt to spot key elements of re-useability that might fail to be achieved if not reviewed early in the design process - the success of this exercise remains to be proven.

THE FUTURE

This project has been immensely valuable in highlighting problems and pointing the way to solutions. In Ada terms it has been instructive to tread our way through what is still an immature market. There has been no major problem in using Ada, but there is considerable need to limit the use of Tasks and Generics to areas in which the 'costs' do not exceed the 'benefits', and undoubtedly there are 'real-time' costs for the unwary.

We will be looking very carefully at the Design Tool market - the 'ideal' AdaPDL design tool could merge some of the Smalltalk Browser features with the ability to define abstract data types and directly generate Ada code from checkable AdaPDL, provided there is no undue need for complex graphical representations.

Also at the Requirements Analysis end of the lifecycle there is scope for a simplified tool, not dissimilar to that required for design but without the rigour of AdaPDL, with Traceability through to the Specification documents achievable in a 'text only' environment (Hypertext possibly?).

With the correct tools in place to steer and control the shape of the Analysis and Design processes, the Ada skill area could be focussed onto a smaller and more specialized team to handle the effective mapping, or re-mapping, of the AdaPDL solution onto code during the implementation phase.

In the absence of the optimum tools, the above philosophies can be implemented in the paper and pencil domain but this involves an increased need for control during the Analysis and Design processes.

If we manage to achieve any significant proportion of our goal of producing a kernel of re-useable, well documented and highly encapsulated code, we will need to establish better machinery for making its existence known across projects; the control and dissemination of this 'high value' code is a problem yet to be fully addressed.

The future existence of significant amounts of re-useable code poses its own problems of security - we were previously protected by the 'its easier to rewrite than to re-use' attitude spawned by 'distributed' code. However with re-useable 'objects', clearly defined and highly encapsulated, the code suddenly has high re-use value both internal and external to the Company - code security now becomes doubly important.

IN CONCLUSION

1. Ada offers stability and sufficient of the good things that it acts as a catalyst to the triggering of more important fundamental concepts of software design.

2. OOD is a very effective Design Mechanism for simulation.

3. Encapsulation and defined 'inter-object interfaces' is well handled by OOD and a critical and necessary feature in the transition to re-useable code.

4. Code re-use is essential for survival in today's marketplace.

5. Ada is complex and requires careful and experienced use to avoid 'real-time' pitfalls.

6. Requirements Analyses and Design Tools using AdaPDL should in future provide optimum environments in which the majority of the Design team can work, remote from detailed Ada.

7. Semi-automatic (Steered mapping) from AdaPDL to direct Ada code generation should allow a small team of Ada skilled users to support a much larger team of Ada applications designers.

8. Entry costs are high for compilers, support tools, workstations and training.

9. A high level of training is desireable if the potential benefits are to be realized.

REFERENCES AND SUGGESTED READING

[1] Pressman, "Software Engineering - A Practitioner's Approach", Second Edition, McGraw Hill.

[2] Peterson, G.E., "Object Oriented Computing", Computer Society of the IEEE.

[3] Somerville, I., "Software Engineering", Third Edition, Addison Wesley.

[4] HOOD Manual, Issue 2.2, European Space Agency.

[5] Booch, Grady, "Object Oriented Development", IEEE Transactions on Software Engineering, Feb. 1986.

[6] Welch, P.H., "A Structured Technique for Concurrent Systems Design in Ada", Proceedings of Ada-Europe International Conference, Edinburgh 1986, Cambridge University Press.

ABOUT THE AUTHOR

Peter Baker is a Consultant Engineer with Marconi Simulation, having more than 20 years experience in the development of software, mainly in real-time training simulators. He is currently serving as the lead engineer on Marconi Simulation's drive to enhance software engineering techniques for military simulators and trainers, this work building on previous experience in designing and implementing sensor simulators for Acoustics, Radar and Electronic Surveillance Measures for a wide range of military and commercial trainers.