Software Engineering Techniques in the CERN RD-38 (CICERO) Project

R McClatchey[1], J-M Le Goff[2], N.Baker[1], R Barillere[2], C Escrihuela[2], E.Bernard[3]

[1]Dept Of Computing, Univ West of England
[2]ECP Division, CERN
[3]SPACEBEL Informatique

Abstract: The complexity of the software required in the forthcoming control systems for LHC experiments necessitates a re-think of the strategies employed in the design of such systems. Use of software engineering standards and systems development methodologies are required to ensure the success of High Energy Physics (HEP) control system design. The CERN RD-38 (CICERO) project aims to use modern software engineering methods such as Object Orientation to design the main building blocks of a generic control information system which will be based on the distributed object standard, CORBA [1]. This paper outlines conclusions that can be drawn from the use of Object Orientation and the ESA standards in the design of Cortex [2], an integrating environment which enables user control objects to be 'plugged and played' in CICERO.

1 The Cortex approach to Designing HEP Control Systems

The large number and variety of parameters that require monitoring in HEP experiments as well as the complexity of the software to perform that monitoring and control has forced systems designers to reconsider the entire design of HEP control systems [3]. The increase in the numbers of sensors and in functionality required by these experiments has led inevitably to the construction of very heterogeneous and distributed control systems where integration and communication between the different detector (sub-)systems has become an issue. It is only by reducing the apparent complexity of the control system that the operation of these systems and their overall maintenance can be efficiently handled by physicists. In addition, since these experiments evolve over time, any HEP control system must also be scalable and flexible to change and provide reusability of its constituent components.

To help reduce development costs and to simplify the apparent complexity of the HEP control systems, physicists are looking for partially reusable solutions to their technical problems. However, in the recent past it has proved difficult to integrate commercial products together (such as PLCs with front-end VME crates) and to integrate these products with existing CERN-made control (sub-)systems [4]. In essence, what is required by those responsible for HEP control systems is an overall framework to facilitate integration between control system components. Such a framework (or software integration platform) should go beyond defining standard interfaces; its design should guarantee that commercial products and 'home-grown' systems can exchange information and collaborate regardless of the organisation of the overall control system. The CICERO research and development project, involving both academic institutes and industrial partners, aims amongst other things to investigate the use of Object Oriented Design (OOD) techniques and distributed system standards in providing an integration framework (Cortex) to simplify HEP control systems design. The Cortex design concepts are described in another paper contributed to this conference [5].

Experience of the development of control systems for the LEP experiments [3] and anticipation of the increased demands which will be generated by the new and larger experiments for LHC, has identified the main constraints for the design of such an integrating scheme. Firstly, since the LHC experiments will be equipped with > 100,000 sensors and activators, the *management of control system complexity* is a major consideration. Object-oriented design techniques embodying abstraction, inheritance and the use of classes and objects will aid the design here. Secondly the problem of *concurrent and collaborative software engineering* requires addressing. This is particularly true when the development of the control software is carried out by engineers who are separated geographically. Thirdly, the software developed should provide *stability, flexibility and availability* of control system elements so that system downtime (both for repair and for upgrades) is minimised. Finally, the control system software should provide *balanced, distributed processing* in the heterogeneous environment of High Energy Physics (HEP) control systems.

The architecture of any distributed control system will inevitably change during the lifetime of the accelerator or the experiment it is operating. As a consequence, Cortex must support a mechanism to allow a new version of the distributed control system to be in preparation off-line while an older one is operated on-line. Furthermore this off-line/on-line facility is needed since tests and validation (and possibly simulations) of a new configuration will be needed before it is applied to the operating on-line system. These constraints have led to a so-called dual-face approach being taken in CICERO for the Cortex integrating framework:

* an off-line Cortex Repository is proposed to handle the logical descriptions of the architecture of the distributed

control system and to describe the various information and commands to be exchanged between the different components.

* an on-line Cortex Infrastructure is proposed comprising so-called Distributors through which the User (control) Components can exchange information and commands in a pseudo- 'plug-and-play' fashion. A generation mechanism is provided to facilitate updates of the on-line Infrastructure according to the Repository contents.

The physicist then builds a logical model of the required control system architecture off-line in the Cortex Repository, specifying detail at the configuration level to enable updates to be carried out on the existing architecture. These updates are validated against the existing architecture and the generation procedure instantiates the updates and creates the new updated Cortex Infrastructure. The details behind the operation of Cortex have been dealt with elsewhere [6] & [7]; this paper examines the design approach taken in implementing the building blocks behind Cortex. In particular, section 2 investigates constraints on the Cortex design approach and section 3 outlines the methods followed in delivering a Cortex prototype. A critique of this approach is presented in section 4 and conclusions are drawn in the final section.

2 The Need for Software Engineering in the Design of HEP Control Systems

Historically, high energy physicists have been slow in adopting the software engineering methods of computer scientists. The systems developed in HEP have often responded to the development needs on an ad-hoc basis, with many short term solutions to similar problems [3]. Structured software development methods have been used [8] for the analysis and design of control systems for HEP experiments at LEP. Since then, experience of structured software engineering methods and CASE tools has increased at CERN [9], [10]. As yet, however, there have only been a few examples of the use of OOD methods in the design of control systems at CERN. One notable exception is that of Myers et al [11] which investigated the use of OOD in a controls environment. In industry, OOD techniques are more prevalent and have been used in the design of control systems for Supervisory Control And Data Acquisition (SCADA). Ericsson et al [12], advocate both the use of object design techniques and the use of CORBA as the distributed communication platform for control systems. The CICERO project is using both OOD and CORBA to implement a generic control information system for LHC experiments.

As HEP control systems evolve for LHC, the need for rigor in software engineering increases in importance. In addition, as the development of these complex systems is increasingly carried out remotely from CERN or by developers on short-term contracts at CERN, the need for clearly defined deliverable code and interfaces between (sub-)systems also becomes crucial. As a consequence, software engineering standards have been investigated in the last few years by large groups of developers in HEP. Furthermore, work at the European Southern Observatory [13] into standards has recommended the use of the European Space Agency software engineering standards alongside those from the IEEE and IEE.

LHC experiments will involve collaborations of many tens of institutes and over 1,000 physicists, engineers and computer scientists from around the world. The knowledge required to construct and monitor the hearts of the experimental detectors will be distributed between these institutes making it difficult to impose standards. There will consequently be significant problems of information transfer in ensuring that each detector subsection retains autonomy of control but can work with other detector subsections for data-acquisition. In addition, the LHC detectors will be required to have a long lifecycle, since the experiments will take data for several years and as a consequence maintainability will be an important consideration. Clearly, to reduce problems of interoperability and integration the project needs to adopt a set of standards both for the development of the CICERO software and its operation across heterogeneous hardware platforms [5]. The methods used and the standards followed must enable collaborative working in a systematic manner and adherence to a strict discipline. Support tools are required to provide cross-checks of the design and to reduce the possibility of error in the construction of design models.

3 The Choice of Methods and Standards in CICERO

The European Space Agency Procedures, Specifications and Standards [14] method has been atapted as an essential feature of CICERO. This standard has been developed for the European Space Agency to ensure that any project has the best chance of being successful. There are two major parts: the product standards themselves and the procedures standards used to produce them. The products are the documents and software used to create, use and maintain software. The procedures guide the system developer in software project management, software configuration management, software verification and validation and software quality assurance.The documentation for this standard includes both mandatory and optional sections, divided into three levels. In order to meet the ESA standard, all mandatory operations must be carried out or documentation produced as appropriate. The process of production is divided into six phases, following the standard life-cycle model of User Requirements Definition, Software Requirements Definition, Architectural Design, Detailed Design and Production, Transfer and Operations & Maintenance. In addition there are documents on Structured Analysis, Fortran Coding, Ada Coding & C Coding standards.

Management of the software lifecycle is catered for in the ESA standards through the use of a Software Configuration Management Plan (SCMP), a Software Verification and Validation Plan (SVVP) and through Software Quality Assurance (SQA). These plans are detailed in [14] and provide the project manager with requirements for identifying, controlling, releasing and changing software releases and for recording their status. The SVVP provides for the review, testing and auditing of the delivered software products. Further, the project manager can ensure quality is being maintained in software delivery by following the recommendation of the SQA plan.

The ESA standards were originally based on the ‘Waterfall Model’ of the software lifecycle, following a phased approach to software development. Modified forms of this approach include the incremental delivery and evolutionary development models. In the development of Cortex an *evolutionary* approach has been adopted which overcomes some of the limitations of the waterfall model (eg implementation only in full and at project completion). The evolutionary approach allows for the planned delivery of multiple releases of Cortex, with each release incorporating the experience of earlier releases in a manner analogous to the ‘Spiral Model’ of software development suggested by Boehm [15]. Given the often speculative nature of the research in CICERO and the mechanisms of staged funding, following the evolutionary approach to systems development reduces the inherent risk of project failure.

The ESA standards do not prescribe a method to follow in designing software systems, although they promote a functional decomposition approach to systems design. However, the nature of HEP control systems - their complexity, the distribution of components across heterogeneous systems and the set of fundamental system services required in these systems (such as replication, migration and failure and error recovery) point to the use of an object encapsulation approach as an appropriate mechanism of hiding and managing system complexity. Such objects can improve inter-operability due to their semantic richness and can be combined in novel ways to promote information processing and minimise code modification. Furthermore, adopting an Object Oriented approach in design facilitates the encapsulation of existing non-object based (legacy) systems so providing software reuse and a clear upgrade path for control systems designers. Consequently, an Object Oriented approach has been adopted in CICERO.

In the development of CICERO, Object Oriented programming languages (eg C++), Object Oriented Databases (eg Gemstone, O2) and object-based distributed systems standards (CORBA [1]) are being investigated. In integrating such technologies, an object oriented design approached is required to be followed which is sufficiently mature, academically sound and supported by available Computer Aided Software Engineering (CASE) tools. Foremost amongst these is the method developed by James Rumbaugh and colleagues, OMT [16]. The OMT method comprises a number of models which are developed and enhanced as the project moves from requirements analysis through design to implementation. OMT comprises three models and four software development stages: the Object, Dynamic and Functional Models and the phases of Analysis, System Design, Object Design and Implementation.

Of widest use on the CICERO project have been the Object/Class model of OMT(a form of Extended Entity Relationship model showing static data relationships in the software), event traces (showing sequences of messages sent between objects to accomplish a given function) and the Dynamic Model (State Charts to define the temporal ordering of events impinging on a given object). The Functional Model which encompasses data-flow diagrams and function definitions has not been used much in CICERO. The Object and Dynamic models have been supported by the use of a diagramming tool, Select initially, and latterly Westmount Case. In addition, the CICERO project has used a code generation tool, OBLOG, which supports the OMT methodology, to generate User Components (in C++) from the OMT Object/Class model. The OBLOG tool integrates concepts from semantic data modelling and concurrent processing, employing objects as a unifying concept and aims at a conceptually seamless methodology from requirements to implementation [17].

4 A Critique of the Use of OMT and the ESA Standards

The software quality assurance process of ESA enforces the classic waterfall model of software engineering by requiring that no phase can begin until the previous one is complete. While Object Oriented Design is referred to in the Guide to the Software Architectural Design Phase [14] and Object Oriented Programming is mentioned in the Guide to the Software Detailed Design and Production Phase [14], the specified method for architectural design is largely a top down approach. Therefore in using OMT with the ESA standards some mapping from the four software development phases of OMT onto the six phases of the ESA standards is required.

CICERO has mapped OMT on to ESA in the following manner: The User Requirements phase of ESA is mapped onto a text-based definition of the problem statement in OMT. The Software Requirements then maps directly onto the Analysis phase in OMT where an understanding of the application domain is achieved and a model of the system is developed. The OMT phase of System Design then matches that of Architectural Design in ESA when the overall architecture of the system to be developed is determined. Next, the Object Design phase of OMT is mapped onto the Detailed Design and Production phase of ESA when the models are consolidated and the final system design is specified in detail. The Implementation phase of OMT corresponds to the Transfer and Operations and Maintenance phases of ESA (see figure 1).

Following the ESA standards and an evolutionary lifecycle model, the Cortex element of CICERO has gone through the first spiral of User and Software Requirements specification followed by Architectural and Detailed Design resulting in a demonstrable prototype. The disciplined use of the ESA methods during the prototype development undoubtedly helped to clarify the design of the prototype system and to focus the effort of the designers who were geographically remote (in the UK, Belgium, India, Hungary and CERN). Such a use of the ESA standards does, however, require care, especially in identifying the boundaries between the different phases of the lifecycle. In principle, ESA standards do not enforce any particular methodology. In practice, however, the ESA guidelines clearly support a functional breakdown of the problem statement and hence imply non-object-oriented methodologies. Some trade-offs had to be made to support the OMT methodology, especially in the Architectural and Detailed Design phases of the

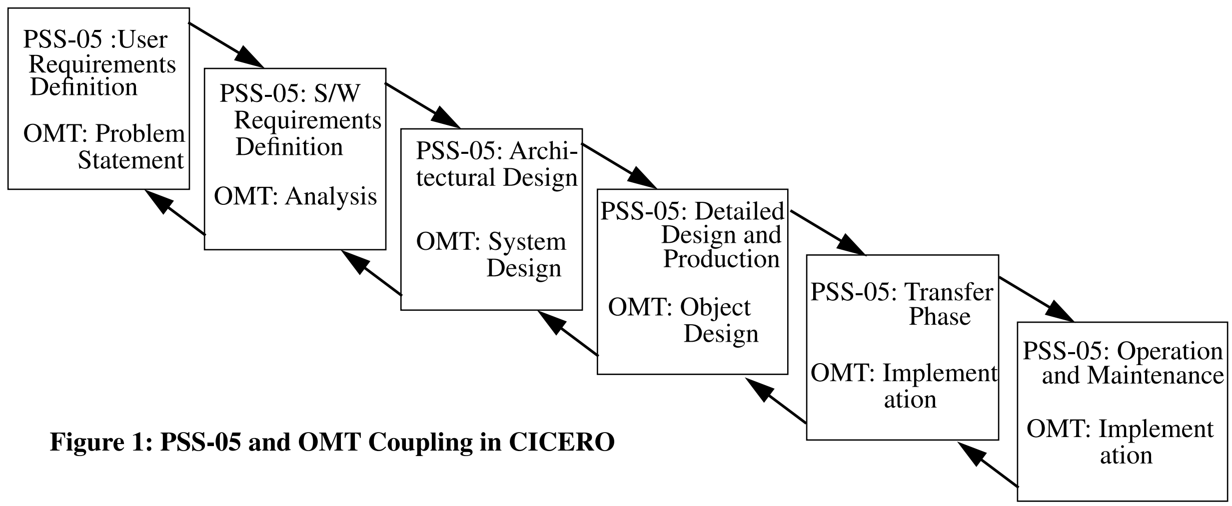

Figure 1: PSS-05 and OMT Coupling in CICERO

ESA standards. One example is in resolving the relationship between the demands on the documentation produced in accordance with the ESA standards, particularly in the prominence given to natural language in the User and Software Requirements documents, and the need for more precise semantics demanded by the OMT modelling approach.

Software produced for any pilot project must be similar to production software with respect to robustness and reliability. Therefore management guidelines supporting the software lifecycle as enhanced by the ESA-PSS 05 standard should be strictly followed. In particular, as stated earlier, a complete set of Software Project Management, Software Configuration Management, Software Verification and Validation and Software Quality Assurance plans should be produced in parallel with the software lifecycle documents. This is often overlooked in the non-software engineering world of HEP and in particular when costs are a major constraining factor. It is the experience of CICERO that these important software management procedures should be addressed in any application of the ESA standards.

The OMT notation itself is strong in describing the abstract models used in object design, but is lacking when describing the fully implemented system. This failing was particularly noticeable during the Architectural and Detailed Design phases of the prototype. Yet it is here, when the volume of design information increases substantially, that notations to capture the design and tools for their manipulation are essential. Work on design notations and the capture and re-use of designs is the focus of some researchers in the OO field and the results of this work should be consulted by groups using the ESA standards with object-oriented design techniques. OMT is not particularly targeted at the design of real-time systems and is not sufficient to handle concurrency. As a result, the treatment of some issues of concern in the development of Cortex have been inadequately addressed by OMT. Some of these issues are better handled by later developments of the OMT approach, notably the Syntropy method [18]. Syntropy provides additional rigour to the OMT method, particularly in the design of cooperating processes. This is a potential area of future research in CICERO.

It was clear during the prototype development phase that in normal discussion about Cortex, designers and users make great use of concrete examples of usage of the system and specific, often important, examples of message sequencing within the implemented system. It is felt that the efficiency of the design process and of communication of the design to the users could be improved by the incorporation of scenarios and use cases into the formal documentation and design process following the Jacobson approach [19]. The most appropriate phase for incorporation of such use cases is clearly the User Requirements specification of ESA. However, these familiar scenarios are likely to become part of the vocabulary of the project team and be refered to at subsequent stages of the lifecycle thereby adding clarity to the design discussion.

The quality of tools support for methods remains an issue for CICERO. Platform specific tools, difficult access to design repositories and the lack of support for multi-user and multi-site development, lead to a loss of efficiency. In particular, this can generate additional conversion and cross-checking tasks. Methods for managing versions of documents, supporting relationships between models from different viewpoints on the design and tracking the process of design decision making are all desirable in the selected toolset. As yet there is no obvious single tool for supporting the use of OMT in developing control systems. Tool evaluation, selection and guidelines in their use in an integrated manner remain tasks for the next phase of the CICERO project. The use of CASE tools like OBLOG for the automated generation of CORBA compliant User Component code will be investigated further, especially to increase the performance of the generated code, to deal with more complex data types and to support concurrent access etc.

5 Conclusions

Since being approved in February 1994, the RD-38 project has grown and continues to attract further commercial and academic research interest. A demonstration prototype has been constructed by the CICERO group to validate the ideas proposed by the collaboration and to illustrate new concepts. This prototype is a small scale illustration of a distributed control system for a real HEP experiment. This prototype is operational and has been used to carry out some basic performance tests. Preliminary results have been obtained, using different user components and distributor configurations and the quantitative results have been reported elsewhere [7].

A clear conclusion to be drawn from the prototyping phase is that standards, while necessary for the structured development of a project, especially involving widely dispersed collaborators, can introduce problems unless they are well integrated with methods and tools. In CICERO their disciplined use has demonstrated that a widely dispersed collaboration can work together even under tight time and resource constraints to produce a functioning system. However, the deliverable-driven approach encouraged by the use of the ESA standards is not entirely compatible with an Object Oriented approach to design and with the prototypical nature of the CICERO research. The prototype has nevertheless been successful in, amongst other things, demonstrating the limits to which the ESA standards and OMT can be used in CICERO and in investigating automatic code generation of Cortex User Components using the OBLOG CASE tool. It has successfully followed a mapping between the waterfall model of ESA and the object oriented design of OMT. Recently the next phase of CICERO has been approved. By mid 1996 the project will build a pilot project in an existing HEP experiment at CERN (L3). This will enable further investigation of the ESA standards and OOD techniques (following the evolutionary development approach [20]) and greater use of support tools such as Select, Westmount and OBLOG CASE tools. In addition, the techniques of Jacobson and Syntropy will provide a fruitful area of further research as identified in the previous section.

Within a year the CICERO collaboration has demonstrated that it is possible to build an heterogeneous distributed control application using OOD techniques, commercial products and to deliver high level functionality like alarm filtering, user assistance and on-line documentation to help the user operate the control system. Much work remains in CICERO Phase II to provide a system which could be used in practice. It is expected that CICERO Phase II will reach this goal and will prepare the ground for a final consolidation phase yielding a set of software building blocks to enable the implementation of both LHC experiment control systems and industrial complex control systems.

6 Acknowledgments

The CICERO project involves the following groups to which the authors extend thanks both for financial support and for technical assistance: BARC (Bombay, India), CERN (Geneva, Switzerland), CIEMAT (Madrid, Spain), IVO International (Helsinki, Finland), KFKI (Budapest, Hungary), OBLOG (Lisbon, Portugal), SEFT (Helsinki, Finland), SPACEBEL (Brussels, Belgium), UID AB (Linkoping, Sweden), USDATA (Dallas, USA), UWE (Bristol, UK), VALMET Automation (Tampere, Finland) and VTT (Oulu, Finland).

7 References

[1] CORBA, The Common Object Request Broker: Architecture and Specifications, OMG Pubs 1992

[2] Barillere R et al., "The Cortex Project: A Quasi- Real-Time Information System to Build Control Systems for High Energy Physics Experiments". Nuclear Instruments & Methods A 352 (1994) pp 492-496.

[3] Barillere R et al., "Ideas on a Generic Control System Based on the Experience of the Four LEP Experiments Control Systems". Presented to the ICALEPCS '91 Conference, Tsukuba, Japan, Nov 11-15 1991, pp 246-253

[4] Dalesio L R et al., "The Experimental Physics and Industrial Control System Architecture: Past, Present and Future", Nuclear Instruments & Methods A 352 (1994) pp 179-184

[5] Barillere R et al., "CICERO: Control Information system Concepts based on Encapsulated Real-time Objects". Invited talk at the ICALEPCS ‘95 Conference, Chicago, USA, October 30th - November 3rd 1995.

[6] Govindarajan G et al., CICERO: 1994 Status Report. CERN RD-38/LHCC/95-15.

[7] Le Goff J-M & McClatchey R., “CICERO RD38: A Distributed Information System for HEP Controls”. Proc of the CERN School of Computing, Arles August 1995.

[8] Charity T., McClatchey R. & Harvey J., "Use of Software Engineering Techniques in the Design of The ALEPH Data Acquisition System"., Computer Physics Communications Vol 45 (1987) No 1-3 pp 433-437

[9] Buono S et al "Software Engineering Techniques and CASE Tools in RD13", Nuclear Instruments & Methods A 352 (1994) pp 383-385

[10] Mato P et al., "The New Slow Control System for the ALEPH Experiment at LEP", Nuclear Instruments & Methods A 352 (1994) pp 247-249

[11] Myers D. et al "Use of Object Oriented Techniques in a Beam-Line Control System", Proc of Int Conf on Computing in High Energy Physics, 21-27 Apr 1994, San Francisco, USA.

[12] Ericsson G et al., "Structured Development of Industrial Control Systems for Power Networks", 2nd IEEE Conference on Control Applications, Sept 13-16 1993, Vancouver, Canada

[13] Filippi G., "Software Engineering for ESO's VLT Project", Nuclear Instruments & Methods A 352 (1994) pp 286-389

[14] ESA PSS-05-02. ESA Board for Software Standardisation & Control (BSSC), 1991

[15] B. W. Boehm, ‘A Spiral Model of Software Development and Enhancement’, IEEE Computer **21** (5), pp 61-72 (1988).

[16] Rumbaugh J et al, Object Oriented Modeling & Design. Prentice Hall 1991

[17] Ehrich H-D., “Fundamentals of Object Oriented Information Systems Specification & Design: the OBLOG / TROLL Approach”, Nuclear Instruments & Methods A 352 (1994) pp 375-378.

[18] Jacobson I., Object Oriented Software Engineering - A Use Case Approach. Addison-Wesley 1992.

[19] Cook S. & Daniels J., Designing Object Systems - Object Oriented Modeling with Syntropy. Prentice Hall 1994.

[20] Mazza C., "Controlling Software Development"., Nuclear Instruments & Methods A 352 (1994) pp 370-374