Contemporary Approaches to Control System Specification and Design Applied to KAON

George A. Ludgate and Edwin A. Osberg, TRIUMF, Vancouver, B.C.
Don A. Dohan, SSC Laboratory, Dallas, Texas

Abstract

Large data acquisition and control systems have evolved from early centralized computer systems to become multi-processor, distributed systems. While the complexity of these systems has increased our ability to reliably manage their construction has not kept pace. Structured Analysis and Real-time Structured Analysis have been used successfully to specify systems but, from a project management viewpoint, both lead to different classes of problems during implementation and maintenance.

The KAON Factory central control system study employed a uniform approach to requirements analysis and architectural design. The methodology was based on well established object-oriented principles and was free of the problems inherent in the older methodologies. The methodology is presently being used to implement two systems at TRIUMF.

I. Introduction

Experiments data acquisition systems and accelerator control systems have evolved from early centralized systems to become distributed, multi-processor systems employing large, complex databases, artificial-intelligence software techniques and sophisticated direct manipulation user interfaces. While the complexity of these systems has increased several orders of magnitude over the last two decades our ability to reliably specify them and manage their construction has not kept pace with the other technological gains.

Text based requirements specifications are known to be too bulky and hide ambiguities and omissions. To address these concerns many graphical, model-based systems development methodologies were created during the late seventies, along with software tools to support their use on large projects. One of these popular techniques, called Structured Analysis (SA) [1], has been used successfully at CERN [2] and another, Real-time Structured Analysis (RT/SA) [3], has been used at TRIUMF [4]. Both suggest the use of Structured Design [5] and structured programming to complete the implementation.

II. Analyzing the KAON Factory Control System Requirements

The KAON Factory Project Definition Study [6] included an investigation whose goal was to produce a preliminary hardware and software design, and a cost estimate for the control system. A contemporary object-oriented requirements specification methodology [7] was chosen for this study to avoid the perceived problems with the use of SA or RT/SA on large system development projects. The methodology embodies a uniform approach to requirements analysis, architectural design, detailed design and coding based on well established object-oriented principles.

III. Criticisms of Structured Analysis

Structured Analysis followed by Structured Design has been used extensively in the business domain, and more recently in particle physics experiments and accelerator control systems. It is a "top-down" approach to system design that starts with the establishment of a Context Diagram. A Context Diagram shows the system to be built (denoted by a circle) communicating with other systems in its environment, called Terminators (denoted by boxes), by flows of data (denoted by directed lines). The new system is viewed as a single process capable of storing and transforming information in response to events in its environment.

An analysis of the system proceeds by developing a second diagram showing a decomposition, or refinement, of the Context Diagram process into several "lower level" processes, linked by data flows. The proposed lower level processes are provided the same external inputs as shown on the original Context Diagram and must produce the same external outputs. The cooperative working of these lower level processes must achieve the same effect as the single process shown on the Context Diagram for the proposed decomposition to be acceptable. Further analysis work proceeds by recursively "refining" each process until the analyst feels a process can be described textually and needs no further refinements.

Several major problems exist with using Structured Analysis on a large project:

- It is "top down". Detailed analysis of low level processes can only proceed after a "high-level" analysis of the system has been completed. Processes remaining to be refined obviously depend critically on refinement decisions made at higher levels. Any major changes to the high level process decomposition of a system may invalidate the analysis of all processes below that level.
- Analysts can only be added to the project as the number of refined processes increases.
- There is often no unique decompositions of a process into lower level processes. Analysts attempt to apply functional decomposition but there are as many

0-7803-0135-8/91$01.00 ©IEEE

understandings of the term "functional" as there are analysts. Thus, the "same" functionality observed by two different analysts may be decomposed differently, forever hiding the common features.

- At any point in the analysis of a new system it is not clear how many more levels of decomposition will be required.
- The *behavior* of a system is hidden "inside" process specifications. Changing the behavior of the system may involve modifying several processes that coordinate their actions through stored data.

Control system engineers employing SA tend to produce refinements of the Context Diagram that are naturally object-oriented; associating one or more (behavior determining) processes and (property specifying) stores with each Terminator [8].

IV. Criticisms of Real-time Structured Analysis

Real-time Structures Analysis [3] is an improvement over classical SA from the perspective of managing a large project. Like SA, the methodology advocates creating a Context Diagram for a new system first. The methodology then requires one or more analysts to model the environment of the new system by finding and recording all events in the environment which effect the system. The system's responses to each event must be planned and recorded along with the event, as must the mode of behavior before and after the occurrence of the event. These analysis tasks can be carried out by several analysts relatively independently of each other and, as long as only one master Event/Response List is maintained (which must also include all event synonyms) no duplication of requirements will occur.

The behavior and process structure of the new system is finally derived from the complete Event List by a process similar to the derivation of a classical control system transfer function from a *complete* specification of its inputs and outputs as continuous functions of time. The designer examines the Event List and constructs State Transition Diagrams (STDs), representing the behavior of the system at relative points in time. No heuristics were originally given to guide this "construction" process [3].

There are several major problems with the use of real-time systems analysis on a large project:

- Experience has shown that even with many analysts working concurrently, it is unlikely they will discover the *complete* Event List for a complex system - there being no recommended approach to partitioning the search for events nor for ensuring that all events are discovered.
- The internal process structure of the new system depends, *in principle*, on only the events discovered and the responses planned. Thus, while it is possible for several analysts to independently search for events and plan system's responses, it is not feasible for them to independently create models of the systems internals. Design work cannot start until the last event is discovered.
- Similarly, in principle, it is not possible to predict the amount of re-structuring of a system's internals required to accommodate the discovery of a *single* new event. In practise, partitioning the Event List by Terminators will limit the "propagation" of changes.

Well known approaches to using this methodology apply an object-oriented partitioning of the "event space" by factoring the search for events by Terminators and associating an STD with each terminator [7].

V. Benefits of the Object-Oriented Approach

A contemporary object-oriented approach to system analysis and design was selected to model requirements for the KAON Factory central control system [9–11]. This approach directs system analysts to initially study the application from the viewpoint of the users. For accelerator control systems the application domain includes operators (one kind of user) controlling accelerators composed of ion sources, magnets, rf cavities, beam diagnostics, production targets, vacuum equipment etc.; all working toward the goal of accelerating and transporting different kinds of beam from the ion source to one or more production targets according to a schedule. Each device type or conceptual entity (e.g. schedule) is considered to be an "object" for the purposes of this methodology and its role in the application domain is, therefore, subject to an object-oriented analysis (OOA).

Two analysis models are developed initially for each object identified to be part of the application; a dynamics model describing behavior and a statics model describing properties. The dynamics model, usually in the form of an STD, represents the object's modes of behavior and the allowed transitions between those modes. Conversely, the statics model of an object identifies both the properties that are needed to completely describe it at any instant in time and its relationships with other objects in the application domain.

Models of KAON Factory sub-systems were created in collaboration with sub-system engineers (experts) and validated for quality control. For example, a beam line magnet may only be in one of the modes {OFF, RAMPING UP, ON, RAMPING DOWN} at a time and no transitions are possible between ON and OFF states without going through one of the ramping behaviors. Similarly, values for the two properties {MAGNETIC FIELD, TEMPERATURE} may be required by the control system at all times to ensure the magnet is operated correctly.

In contrast to SA, the OOA approach is "bottom up". The analysis of each identified object can, in principle, proceed in parallel once the goals of the control system have

been specified. In fact, the development of an object's dynamic and static models can also proceed in parallel, subject to the availability of experts familiar with the object being investigated. The two analyses of an object must be reconciled; discussions concerning behavior illuminate properties while discussions about the objects relationships to other objects will involve discussions of behavior modes.

Unlike RT/SA, the levelled internal structure of an object-oriented system is directly related to, and *derived* from, the statics model, while the detailed (process model) structure is *derived* from the dynamics models. Event responses appear explicitly in dynamics models, partitioned according to the object causing the event. The structural models are, therefore, more stable to change and to the addition or removal of objects from the application domain. Similarly, the list of events to which the control system must respond is *derived* from an examination of all of the dynamics models rather than searched for as in RT/SA.

Both SA and RT/SA use Structured Design to cast their "data-flow" specifications into a hierarchical form more easily adapted to the "subroutine call" form required by structured programming. In contrast, OOA is best followed by Object-Oriented Design (OOD). In OOD the static and dynamic models from OOA are *mapped* onto similar models of a design architecture (composed of processors, processes, inter-process flow and inter-processor flows, classes and messages etc.). The nature and form of the design models are unchanged as is the interpretation of the diagraming notation. No leap of faith is required to produce the design, unlike that required by practitioners of the "art" of Structured Design in going from data-flow diagrams to structure charts.

The OOA/OOD methodology, particularly when followed by coding in an object-oriented language, incorporates the best features of the earlier SA or RT/SA methodologies namely graphical models, simple notation, implementation independence, etc.. However, *logic*, the underlying basis for OOA and OOD, pre-dates all other system development methodologies and provides a firm, well known formal framework within which developers can work.

From a project management standpoint, relating the volume of analysis and design work to be completed to the number of objects discovered in the application domain is a useful metric for establishing the effort, and hence the manpower, needed to complete the two phases. This metric is determined early in the project, at the start of analysis. Similarly, the completion of analysis and design models for *each* object serves as milestones in charting the progress of the project.

VI. Conclusion

OOA and OOD have been presented as solutions to the system development problems associated with the use of the older technology SA or RT/SA methodologies. In addition, the approach leads to a better factoring of development effort and, therefore, to improved project management due to the occurrence of natural units of work.

The KAON Factory control system was designed using these two contemporary object-oriented methodologies and two projects are presently underway at TRIUMF to implement sub-system control systems using this same approach. Both systems will have object-oriented requirements specifications and be implemented using the C language running under the Vsystem product [12].

VII. References

[1] Tom DeMarco, *Structured Analysis and System Specification*, Yourdon Press, 1978, ISBN 0-13-854380-1.

[2] *Data Acquisition and Computing*, CERN/LEPC/85-9, March 1985.

[3] P.T. Ward and S.J. Mellor, *Structured Development for Real-time Systems*, Yourdon Press, 1985, ISBN 0-917072-51-0.

[4] G.A. Ludgate, B. Haley, L. Lee and Y.N. Miles, *The use of structured analysis and design in the engineering of the TRIUMF data acquisition and analysis system*, IEEE Trans. Nucl. Sci., NS-34, pp. 157-161 Feb. 1987.

[5] E. Yourdon and L.L. Constantine, *Structured Design*, Prentice Hall, 1979, ISBN 0-13-854471-9.

[6] *KAON Factory Study: Accelerator Design Report*, 1989, TRIUMF, 4004 Wesbrook Mall, Vancouver, B.C., Canada, V6T 2A3.

[7] C. Inwood, *Analysis of real-time requirements using dynamic object models*, Proprietary course notes, Inwood Real-time Systems, R.R. 2, Kinburn, Ontario, Canada K0A 2H0, April 1989.

[8] G. Morpurgo, *SASD and the CERN/SPS run-time co-ordinator*, Proc. ICALEPCS, Vancouver, Canada, 1989, Nucl. Instr. and Meth. A293 (1990) pp. 385-389

[9] C. Inwood, G.A. Ludgate, D.A. Dohan, E.A. Osberg and S. Koscielniak, *Domain-driven specification techniques simplify the analysis of requirements for the KAON Factory central control system*, Proc. ICALEPCS, Vancouver, Canada, 1989, Nucl. Instr. and Meth. A293 (1990) pp. 390-393.

[10] E.A. Osberg, G.A. Ludgate, S. Koscielniak, D.A. Dohan and C. Inwood, *Dynamic object modelling as applied to the KAON control system*, Proc. ICALEPCS, Vancouver, Canada, 1989, Nucl. Instr. and Meth. A293 (1990) pp. 394-401.

[11] D.A. Dohan, G.A. Ludgate, E.A. Osberg, S. Koscielniak and C. Inwood, *Definition study of the TRIUMF KAON factory control system project*, Proc. ICALEPCS, Vancouver, Canada, 1989, Nucl. Instr. and Meth. A293 (1990) pp. 6-11.

[12] P. Clout, R. Rothrock, V. Martz, R. Westervelt, M. Geib, *Past, present and future of a commercial, graphically oriented control system*, Proc. ICALEPCS, Vancouver, Canada, 1989, Nucl. Instr. and Meth. A293 (1990) pp. 456-459.