CERN-PS 84-03 Co

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

19 MARS 1984

CERN PS/84-03 (CO)
24.1.1984

ESTIMATING THE DEVELOPMENT EFFORT FOR A LARGE PROCESS CONTROLS

APPLICATION SOFTWARE PROJECT

G.P. Benincasa, A. Daneels, P. Heymans, Ch. Serre

CERN LIBRARIES, GENEVA

CM-P00059337

Paper to be presented at the IX. Triennial World Congress
of the International Federation of Automatic Control
IFAC, Budapest, July 2 - 6, 1984.

ESTIMATING THE DEVELOPMENT EFFORT FOR A LARGE PROCESS CONTROLS APPLICATION SOFTWARE PROJECT

G.P. Benincasa, A. Daneels, P. Heymans, Ch. Serre

European Organization for Nuclear Research, CERN, Ch-1211 Geneva 23,
Switzerland

Abstract. This paper presents the economics of the application software developed for a large process control system. The structure of the application software will be discussed in both a technical and economical perspective, and assessed with Barry Boehm's development cost estimating formalism. The results will be compared with the in-house estimates and the actual costs.

Keywords. Control engineering computer applications: economics: process control.

INTRODUCTION

CERN has undertaken the conversion of its 28 GeV accelerator complex to full computer control. Until then, this complex was operated through a variety of systems ranging from hard-wired manual controls to several stand-alone computer systems. Nearly all are now replaced by a single integrated computer system involving a network of 20 minicomputers and 50 microcomputers interfaced to the process hardware through CAMAC. Operation is assumed by 5 identical general purpose consoles.

The software of the controls project is classified in two categories: system and application. System software embraces all operating systems, network software, programming languages, etc. A major activity was the development of a high-level language P+, a PASCAL-type language designed for process control (Cailliau, 1983).

The application software extends the system software activities and covers all aspects of process controls ranging from operator dialogues and displays, to process hardware control algorithms.

A significant part of the projects total development effort has been taken by this application software: the objective of this paper is to assess its economics using Barry Boehm's cost estimate formalism (Boehm, 1981).

This formalism takes into account a number of project attributes which require current control application software first to be put into the very specific context of the CERN 28 GeV accelerator process and its controls. Its structure which has been designed in a technical and economical perspective will be described next.

After a brief recall of Barry Boehm's development cost estimate formalism, it will be applied to the current application software project.

THE ACCELERATOR PROCESS AND ITS CONTROLS

The CERN 28 GeV accelerator complex comprises two linear accelerators, a four ring synchrotron booster, a proton synchrotron, an antiproton

accumulator and various beam transfer lines. Its purpose is to provide various high-energy physics experiments and other accelerators on the site, with beams of particles, protons, antiprotons..., in batches of 1 sec. The characteristics of these beams are modified from one batch to another in a repeated sequence called "supercycle" which is programmed by the operation crew to meet the demand of the various experiments and other accelerators. To date there may be as many as 24 beams of 8 different types in a supercycle.

The operators in the main control room can tune concurrently several beams of the supercycle by controlling a large variety of equipment, e.g. power supplies, and monitor an equally large set of beam measurement equipment in any subset of any accelerator, from any of the 5 general purpose consoles. At the same time, local control for tests or experiments is also possible in remote equipment areas.

The new controls system is based on a network of 20 minicomputers and 50 microcomputers interfaced to the process hardware through serial CAMAC. Though every minicomputer has its individual area of control, one distinguishes essentially three categories: the console driving computers, one per console, front-end computers, one per process, and general purpose computers such as the central synchronizations and beam sequencing system, the message handler, etc. The microcomputers are located in the serial CAMAC crates as auxiliary crate controllers whenever required for hard real-time activities or local data reduction.

APPLICATION SOFTWARE PROJECT AND ITS ENVIRONMENT

General Requirements

All accelerators of the CERN 28 GeV complex have been built between 1957 and 1980 with different technologies. Whereas the earlier machines had full hard-wired manual control, the 800 MeV Booster was first to have most of its controls centralized in an IBM 1800. This pioneering experiment within CERN in 1972 prompted a number of new accelerators or beam transfer lines to be controlled by some ad hoc stand-alone computer

system which was accessible from the main control room. The operation crew has thus a hands-on experience of 10 years with computer controls. The variety of systems gave them the opportunity to evaluate their merits and to establish accurate specifications for the operational aspects of the new system, especially for the application software (Boillot, 1982).

The main requirements which directed the design of the application software, were primarily concurrent and conflict-free controls of any part of the accelerators: high reliability in the sense that some degradation of the control system should be tolerable before it is felt by the beam (beam loss due to system failure $\approx 1\%$): short response time to operator interaction, i.e. "immediate" response (within ≈ 0.5 sec) to requests not synchronised with beam events nor involving very large sets of process equipment: high flexibility to add new facilities for experimental purposes.

At the process end, requirements are not less exacting. Changing the beam characteristics from one batch to another implies that out of approximately 35'000 process variables (Cailliau and co-workers, 1975) several thousands have to be set to different values in time windows as short as 30 msec and that the beam characteristics have to be measured and displayed for the proper batch at a rate of 200 data every second.

Application Software Layout

The application software has been structured in hierarchical layers of highly independent modules (Myers, 1975) which are executed under the supervision of managerial tasks (Benincasa and co-workers, 1978).

There are six layers of modules: the "interface module" (IM), which executes standard control protocols for an equivalent computer-process CAMAC module; the "equipment module" (EM), which includes the controls for every type of hardware equipment, e.g. power supplies, vacuum pumps....: the "composite variable module" (CVM), which controls beam physics variables (Daneels and co-workers, 1984); the "process module" (PM), which executes control functions on part of the process in conditions which have been dictated by the operator during his man-machine dialogue

performed by the "operator module" (OM). Finally, there are specific tasks in the auxiliary crate controllers for severe real-time controls and data reduction.

The "Skeleton"

The modules in most layers of the application software hierarchy run under supervision of appropriate managerial tasks which are responsible for the proper execution of the requested modules. They start and stop the execution of modules, allocate resources, prevent conflicts, ensure synchronisation with specific events at the occurrence of the right beam, dispatch messages throughout the network, check and handle errors.... These various tasks make up the application software "skeleton" as an extension of the computer operating system. They embrace all activities which, in normal circumstances, every application program would have in common. The "skeleton" and associated application modules constitute the basic structure of the application software. All control functions are constructed along that model.

Basic Operational and Controls Facilities

According to the model mentioned above consisting of "skeleton and application module", a number of basic operational and controls facilities have been provided. They are called basic as they apply to every accelerator of the complex as a result of a standardization of the CAMAC interface hardware, of some equipment as crucial as power supplies with their associated analog function generator and timing units, and of a number of operational aspects.

The basic control facilities include the batch-to-batch modification of the beam characteristics, the controls of interface modules, power supplies, analog function generators and timing, setting-up sequences, etc. The basic operational facilities comprise analog signal observation, video signal observation, knob controls, surveillance and alarm reporting, etc...

The "skeleton" and these basic facilities build up a controls kernel which allows at once some substantial operation of an accelerator.

Development Tools

They extend the operating system services and are implemented to provide special-purpose support for the development of specific application modules. They include special-purpose editors, software templates, dictionaries, listers, databases, etc. In addition are the project management aids to keep track of all software modules and to monitor continuously the progress of the applications project.

Specific Applications

Beyond the three categories mentioned in the previous paragraph, extend all specific applications for the various accelerators and in particular for their central synchronization and beam sequencing system. These specific application modules are mainly developed as subroutines, embedded in the skeleton.

Estimating the Application Sub-projects

The application project is hence subdivided into four sub-projects: the skeleton with its 20 modules, 108 basic controls and operational modules, 32 development tools and eventually, to date 747 specific application modules. The latter are composed of 79 modules for the central synchronization and beam sequencing system, 252 for the 800 MeV Booster and 189 for the synchrotron. Another 227 modules are still partly under development for the antiproton accumulator and various beam transport lines. The size of the modules is an average of 400 instructions, which corresponds in general to a one-month full-time development activity for one person.

All modules IM, EM, CVM and PM and real-time tasks in the auxiliary CAMAC crate controller are implemented in assembler or structured assembler language N-PL. The OMs were originally written in the BASIC-like interpretive language NODAL before availability of P+. Now most OMs and PMs are implemented in P+, their average size stays, however, around 400 instructions.

Constraints

It was known from the start that the project would be heavily constrained. Software in general and application software in particular suffers from not being a long established engineering discipline. Its lack of proper cost evaluating techniques that had won their spurs in convincing customers of the legitimate cost of software development makes that any fair effort and schedule estimate is regarded with a lot of scepticism.

It was imposed that the control system be implemented without impeding the normal production run of 24 hours a day of the accelerators. Hence the controls hardware could only be grafted onto the process during the short accelerator shutdown periods which are planned for preventive maintenance. The application programs to drive the new controls hardware should therefore be ready "at the same time" so that the accelerators could resume their running with a new control system.

Further, during its implementation new accelerator projects came up which were given higher priorities and drained resources. The personnel the organisation could release for the application project was therefore not commensurate with its size.

COST ESTIMATE FORMULAS AND DEFINITIONS

The cost estimating formulas which have been applied in this study are based on the so-called intermediate constructive cost model developed by B. Boehm (Boehm, 1981). This model has been selected as it is claimed to have been applied successfully on a fairly wide range of software projects. Their actual cost has been approximated within 20%, 70% of the time.

The intermediate cost model starts with estimating a nominal effort based on the estimated number of software instructions to be developed and the development mode. Next, the nominal effort is weighted against effort multipliers to yield the effective effort. The effort multipliers rate the project with respect to its product, computer, personnel and implementation attributes.

The Nominal Effort and Schedule

The nominal effort is estimated by

$$E_{\text{nom}} = a (K)^b \quad (1)$$

where E_{nom} is the nominal effort in man-months (MM) and K is the number of delivered source instructions, in units of thousands, including all logical instructions and data declarations but not comments. The term delivered implies all software which in the case of the current controls project, have been delivered to operators, physicists and hardware specialists enabling daily operation and experimenting of the accelerators. Hence ancillary software developed by the software engineers to ease the testing and integration of "official" software packages, are not taken into account.

The project schedule T_{DEV} is the time in months it takes to develop the software product based on the Rayleigh distribution which is said to give a good approximation to the manpower distribution for software development. It covers all phases from the early product design until the completion of the integration tests.

$$T_{\text{DEV-nom}} = 2.5 (E_{\text{nom}})^c \quad (2)$$

The coefficients a , b and c depend on the development mode of the project: the so-called organic, semi-detached and embedded modes (Table 1).

The organic mode project is in general small (less than 50 K-instructions) with relatively little constraints and relaxed specifications. It is developed by relatively small teams, where every team member is very familiar with the type of project and the environment. Most team members participate in all project phases from the early design onwards.

The other extreme is the embedded mode project which is developed within tight hardware, software and operational constraints. It applies in general to projects which have to cope with a number of unknown aspects as a result of lack of experience or genuinely new requirements and techniques. The initial design is performed by a small team which will be increased at the time of implementation.

The in-between development mode is called semi-detached. It is characterized by the fact that some parts of the project are actually embedded whereas others can be considered as organic. The requirements range from fairly relaxed to very tight and team members have a mixed level of experience with similar projects.

The values for a , b and c are respectively :

TABLE 1

development mode	a	b	c
organic	3.2	1.05	.38
semi-detached	3.0	1.12	.35
embedded	2.8	1.20	.32

The Effective Effort, Development Schedule and Staffing

The effective effort is estimated by formula (3):

$$E_{\text{eff}} = E_{\text{nom}} * \alpha \quad (3)$$

Here α is the product of 15 effort multipliers α_i which weigh the development cost with respect to various project attributes:

- product attributes: reliability, ratio of data versus logic and complexity;
- computer attributes: execution time constraints, main storage constraints, computer volatility and turn-around time;
- personnel attributes: analyst capability, application experience, programmer capability, computer and programming language experience;
- implementation attributes: use of modern programming techniques, use of adequate software tools and required development schedule.

These 15 attributes are rated between very low and very high and they are given ratings α_i by B. Boehm (Boehm, 1981).

The effective development schedule is given by

$$T_{\text{DEV-eff}} = 2.5 (E_{\text{eff}})^c \quad (4)$$

The average staffing results from (5):

$$S = \frac{E_{\text{eff}}}{T_{\text{DEV-eff}}} \quad (5)$$

in units of "full-time equivalent software personnel"(FSP), i.e. the equivalent number of people working full-time on the project, e.g. two people working half-time are counted as one full-time programmer.

THE APPLICATION SOFTWARE PROJECT IN BOEHM'S TERMS (APPENDIX 1 AND 2)

Original In-house Project Estimates

The original estimate for the application software to be developed for the CERN 28 GeV accelerator controls results from an early inventory of all requested functions the system should perform based on the experience with the previous IBM 1800-based control system (Daneels, Serre, 1975). Every function was performed by a single application program which catered for all activities ranging from man-machine interaction to hardware controls. Though a first step towards modularization and standardization was obtained by applying process-control oriented data bases and library routines, most programs were monolithic. They were developed as a small individual one-person project in a familiar in-house organic environment and involved on average around 1K instruction. The development time was in the range of 2 to 3 months, so on average 2.5 months.

The new system was going to embody around 400 functions; i.e. 280 inventoried ones and a 50% addition for new and/or overdue ones. In an environment similar to the one of the original control system, the application software would involve about 400 K instructions requiring a development effort of around 1000 MM.

Boehm's intermediate cost model estimates will now be applied first to the overall application software project and next to its various subprojects as described previously.

Estimates for the Overall Project

The high performance which was expected from the application software, short response time, low failure rate, etc., incited novel structuring techniques and algorithms to be worked out. In addition, there was a need for strict conformance with pre-established operational requirements, process hardware specifications and process controls protocols. Moreover, new hardware and software would be developed concurrently. These various features make the development mode for this project fall in the category of semi-detached.

The effective effort is derived from the nominal effort multiplied by effort multipliers which take various project attributes into account.

Product attributes.

- The reliability of the overall application software is evaluated as nominal: i.e. that the losses which in current projects are understood as loss of beam production, are recoverable. Obviously, some software will need to be of very high reliability, in particular the one to modify the beam characteristics from one batch to another. On the other hand, application programs to display beam characteristics merely mean some inconvenience to the operator when they crash; the beam, however, is not affected.

- The ratio between data base and program instruction is less than 10 and therefore rated as low.
- The complexity is rated very high, and stems primarily from the complexity of the accelerator processes themselves. The modification of the beam characteristics from one batch to another involves microprocessor operation to set the various equipment to their proper control value in the proper time window. These values are kept in a database which is distributed amongst the several microprocessors. Measurement and display programs have to be activated at specific events corresponding to the beam the operator tunes. At the console level, the interactive input resources have to be allocated dynamically to programs whereas the output devices, graphical and alphanumerical screens, are shared by several concurrent display programs (Mérard, Pettersson, Serre, 1984), (Heagerty and co-workers, 1984). Beam measurement hardware should be shared between several users and has therefore to be "multiprogrammed" by software.

Computer attributes. The main storage constraint and the computer turn-around time (i.e. the elapsed time between submitting a job and getting the results back) are rated as nominal. The computer volatility (i.e. the frequency at which e.g. the computer hardware and operating system undergo changes) is low. On the other hand, the real-time requirements (responses to man-machine interaction within .5 sec, display of 200 data per sec) are high and they require optimization of the execution time of the programs. Again, this is an average rating for the overall project and programs which have to perform control settings within windows of 30msec have even more severe real-time constraints.

Personnel Attributes. These are evaluated at the level of the team rather than at the level of the individual. The team comprises a 10 men-nucleus of senior engineers or physicists and technicians, with no specific computer science background, who got into computer controls in its early days. Younger computer scientists, professional programmers, university and technical students joined the project in its development phase for a limited period. Between 1978 and 1982, 80 people with average contracts of 1 year, took part in the application software project. After a 4 month education and under guidance of the "nucleus", they were able to produce application packages. The experience of the senior team members and the professionalism of the younger ones, their general ability to communicate and cooperate and their thoroughness enabled them to operate as a high-rated analyst and programmers team, despite its diversity and turnover. On the other hand, the very specific character of particle accelerators in general and more specifically of the CERN 28 GeV complex and its controls, the short-time contracts and the education effort are responsible for the low-rate application experience of the team. Language experience is rated "nominal": professional programmers have nowadays experience in most standard languages, but of course not in the CERN developed high-level language P+ and interpretive language NODAL used for some not time-critical applications. The high turnover of the team again and the specificity of the computer brand, NORISK DATA, gives the team on average a low-rated computer experience.

Implementation attributes. The size and the span of the application software to be developed, urged the use of modern programming techniques: top-down analysis and design, structured programming, chief programmer teams. It was also anticipated that

significant savings could be obtained if appropriate programming tools were provided. Hence, a not negligible effort has been invested in producing editors, frameworks, dictionaries, listings, and databases. NODAL was particularly useful for writing off-hand ancillary programs to aid integration tests. Project management tools were developed to inventory all programs, to evaluate their effort and priority and to monitor their progress. The project is thus being developed in a high-rated environment of programming practise and use of ancillary tools.

Development schedule. Contrary to commercial organizations, a scientific one can accept a relative discomfort because of package size reduction if this could result in reduced staffing. The schedule is therefore rated as nominal.

Resulting effective effort estimate for the overall project. The various effort multipliers α_i are summarized in Appendix 1.

The effective effort E_{eff} , development schedule $T_{DEV-eff}$ and staffing S are tabulated in Appendix 2.

Cost Model Estimates for the Application Subprojects

Because of the skeleton's key position in the control system and the general operational reliance on the basic facilities, their design objectives are primarily speed, reliability and integrity. Failure in any of those aspects propagates at once throughout the overall system which is dragged into poor performances and loses operator's confidence. On the other hand, any optimization in these areas is of immediate benefit to the overall system and operational efficiency. The skeleton is therefore to be developed in an embedded mode, whereas the basic facilities which effectively benefit from the skeleton is more of a semi-detached type. The design is done by a small team impregnated with the overall objectives. Skeleton and basic facilities, i.e. the controls kernel, are provided as prototype to allow the performances to be evaluated and corrected in time, so that it comes to maturity before it is launched on-line. Their various project attributes are similar except for the development schedule which is for obvious reasons much higher for the skeleton than for the basic facilities.

The development tools on the other hand can be considered as many small individual projects of the organic mode. Though their reliability needs to be high, they are less complex, just because of their individuality, than the kernel functions. The same reason makes them particularly suitable for young professional but temporary programmers. They are not dramatically constrained by execution time but need to be available well in advance before the implementation of specific process applications can start.

Amongst the specific process application, the synchronization and beam sequencing of the overall system fills a particular place. Most of the project ratings are therefore identical except for the required development schedule which falls in between the skeleton and basic facilities. Indeed, the synchronization and beam sequencing rely on a number of skeleton functions, and in turn, the basic facilities rely on the synchronization system. The specific accelerator application modules have in general less stringent constraints. Since most critical activities are

ensured by the skeleton, they become less complex and their reliability is nominal. They can be developed as small individual organic mode projects which - thanks to various development tools - makes them particularly suitable for the less experienced programmers.

The effort multipliers for the various subprojects are given in Appendix 1: Appendix 2 gives their respective effective effort E_{eff} , schedule $T_{DEV-eff}$ and staffing S .

Comparison of the "in-house" Estimate with the Cost Model Estimates and Real Expenditures (Appendix 2)

As a result of the breakdown into subprojects, the total development effort is the sum of the ones for every subproject. The real expenditure E_R exceeds the original 1000 MM estimates by nearly 15%: 5% on corrections due to misinterpretation of specification and errors which insinuate themselves in the various development phases; 10% due to modification as a result of changes in the early specification or operational experience, supplementary requests and performance optimization. However, the cost model estimate is very close, i.e. within the claimed 20%, to the original in-house estimate, and remarkably close to the real expended effort !

Similar conclusions can be drawn for most subprojects. A number of skeleton modules and basic facilities, at their prototype stage implemented in the NODAL interpreter language, have been rewritten in P+ to improve their performance and reliability. This new version does account for most of the 15% modification, mainly of the skeleton. It should be pointed out, however, that these around 35 MM of additional effort, i.e. 3% of the total one, were to the general benefit of the global system.

The synchronization and beam sequencing show some 50% discrepancy between the original estimates and the actual effort spent. This is not only explained by the 15% correction and modification but above all by substantial growth of requirements during the project. At that time, the whole accelerator complex was undergoing profound modifications: an entirely new accelerator (for antiprotons) was being added and intricate new beam exchange schemes were being implemented. More than any other subproject, this part is responsible for the extreme beam production versatility of the accelerators. The 28 GeV accelerator complex was entering an entirely novel experimental activity heavily dependent on its synchronization and beam sequencing system.

Finally, the estimates for the booster and synchrotron specific application are again within the 20% limit, whereas the other ones are either nearly terminated and not subject to modification yet, or still under development.

The subdivision into subprojects allowed some of them to be developed concurrently depending on priorities and available resources. In broad lines, between April 1978 and February 1979, the skeleton, part of the basic facilities and the synchronization and beam sequencing were constructed as a prototype package. This activity tapered then off to the benefit of the booster specific applications which were ready end of 1980. The necessary correction and modification were implemented next together with the synchrotron specific applications, which were completed in February 1983. The average equivalent full-time personnel was kept constantly to 20.

Again the cost model estimate for development schedule and average staffing correspond rather well with the "in-house" ones.

CONCLUSION

Two types of conclusion can be drawn from this paper.

First, it can be seen as a case study of Barry Boehm's formalism applied to a large process control application software project. It shows that in case the overall project is broken down into subprojects, the cost model estimates corroborates the in-house ones, or vice-versa. Most in-house evaluations and decisions made throughout this project resulted from personal guidelines based on experience, common sense and intuition: Boehm's formalism puts them in a more rigorous perspective providing a base for improvement. In particular the effort multipliers rating the various project attributes provide a very useful tool for understanding the software trade-off and to establish an efficient development strategy.

Secondly, this paper describes the structure of the application software not only in a technical but also, and maybe mostly, economical perspective. It shows in particular that the subdivision into skeleton, basic facilities, development tools and specific applications is not only morphologically more cost effective, but also intrinsically because the design objectives and implementation strategy of every part can be adapted to their specific purpose. Correction and modification could therefore be confined and their cost kept to a minimum. Similarly, the performances of the overall system could be enhanced significantly at the cost of a small percentage of the overall investment. This subdivision lends itself also particularly well to prototyping. The behaviour of the overall system can be measured early in the project's implementation phase, and it can come to maturity before it starts its life as an on-line system. The high reliability which is illustrated by a "beam availability" of around 99% right from the beginning, is to a major extent due to this structure. Finally, it is also particularly suited for implementation by a highly diversified personnel with a high turn-over.

REFERENCES

- Cailliau, R. (1983). Early experience with the programming language P+. Proceedings, ACM 7th International Computing Symposium, Nürnberg.
- Boehm, B.W. (1981). Software engineering economics. Prentice Hall Inc., Eaglewood Cliffs, New Jersey, USA. Chap. 8, pp. 114-144.
- Boillot, J. (1982). Spécifications de l'équipe "Aspects Opérationnels" pour le nouveau système de contrôle du CPS. PS/OP/Note 82-7.
- Cailliau, R., A. Daneels, L. Mérard and Ch. Serre (1975). Number of Process Variables of the CPS Controls. MPS/CCI/Note 85-58.
- Myers, G.J. (1975). Software through composite design. Petrocelli/Charter, New York.
- Benincasa, G.P., A. Daneels, P. Heymans, Ch. Serre and P. Skarek (1979). Design goals and application software layout for the CERN 28 GeV accelerator complex. 2nd IFAC/IFIP Symposium SOCOCO, Prague, June 1979.

Daneels, A., E. Malandain, M. Martini and P. Skarek (1984). Standard software modules for equipment and composite variable control. PS/CO/Note 84-01.

Daneels, A. and Ch. Serre (1975). Catalogue of functions. MPS/CCI/Note 85-34.

Mérard, L., Th. Pettersson and Ch. Serre (1984). Concurrent execution of real-time displays. PS/CO/Note 84-02.

Heagerty, D., P. Heymans, J. Kenaghan and Ch. Serre (1984). Interactive control of the CERN proton synchrotron complex. IFAC 1984, Budapest.

Appendix 1 Effort multipliers rating of the overall application software project and its subprojects

		R A T I N G S						
		Overall Project	Kernel		Develop. Tools	Specific Application		
			Skeleton	Basic Operational & Control Facilities		Synchroniz. and Beam Sequencing System	Acceler. Process	
P R O J E C T	Reliability	a ₁	1.00	1.15	1.15	1.15	1.15	1.00
	Data/Logic	a ₂	.94	.94	.94	.94	.94	.94
	Complexity	a ₃	1.30	1.65	1.65	1.30	1.65	1.00
C O M P U T E R	Execution time	a ₄	1.11	1.30	1.30	1.00	1.30	1.00
	Main storage constraint	a ₅	1.00	1.00	1.00	1.00	1.00	1.00
	Computer volatility	a ₆	.87	.87	.87	.87	.87	.87
	Computer turn-around time	a ₇	1.00	1.00	1.00	1.00	1.00	1.00
P E R S O N N E L	Analyst capability	a ₈	.86	.86	.86	.86	.86	.86
	Application experience	a ₉	1.13	1.00	1.00	1.13	1.00	1.13
	Programmers capability	a ₁₀	.86	.86	.86	.86	.86	1.00
	Computer experience	a ₁₁	1.10	1.10	1.10	1.10	1.10	1.10
	Program. langug. experience	a ₁₂	1.00	.95	.95	1.07	.95	1.07
	I N F L U E N C E	Use of modern prog. techn.	a ₁₃	.91	.91	.82	.91	.91
Use of software tools		a ₁₄	.91	.91	.83	.91	.91	.83
Required develop. schedule		a ₁₅	1.00	1.10	1.00	1.04	1.04	1.00

Appendix 2 Comparison of "in-house" and cost model estimates, with actual expenditures

		Overall Proj.		Kernel			Specific Application					Sum for Sub-projects
		Original Estimate	Actually Delivered	Skeleton	Basic OP+CD Facilities	Development Tools	Synch. Beam Sequ. System	Booster	Synchrotron	Beam Transport	Anti-protons	
No. of modules		= 1000	907	20	108	32	79	252	189	99	128	907
K		400	330	8	38	13	28	88	67	40	51	330
E I S N T I H M O A U T S E	E O (MM)	1000		50	128	45	92	217	234	142	93	1002
	E (MM)	1129		64	150	56	140	258	235	120	106	1129
	T (MO)											
	DEV					25			18	10		
	S (FSP)					20			20	15		
a												
C E O S T I H M O A U T S E	Devel. Mode	Semi-deta.	Semi-deta.	Embed-ded	Semi-deta.	Organ.	Semi-deta.	Organ.	Organic	Organ.		
	Eeff(MM)	2192	1767	61	165	48	166	260.5	195	113	146.5	1175
	T (MO)											
	DEV-eff	37	34	9	15.5	11	15	21	18.5	15	16	
	S (FSP)	60	52	7	12	4	11	13	11	8	9	
a												

Key : K : number of delivered instructions in units of thousands
 E0 : original in-house estimate in MM
 ER : effort actually spent in MM
 ZDEV : time in months actually spent for development
 Sa : actual staff in Full-time Software Personnel (FSP)
 Eeff : effective effort in MM according to Barry-Boehm's formalism
 TDEV-eff : schedule in months according to Barry-Boehm's formalism
 S : staff in FSP according to Barry-Boehm's formalism