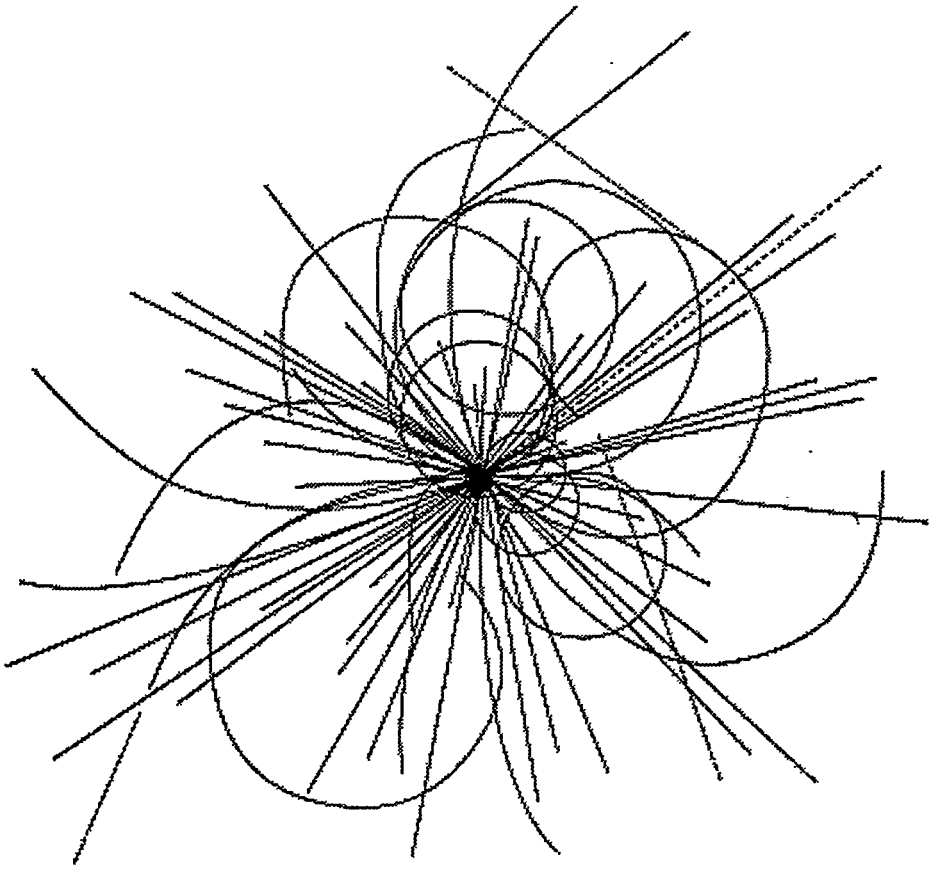
SSCL-Preprint-542

SSCL-Preprint-542
December 1993
Distribution Category: 414

J. Burton
L. Cormell

Physics Research Integrated Development Environment (PRIDE)



Superconducting Super Collider Laboratory

APPROVED FOR RELEASE OR PUBLICATION - O.R. PATENT GROUP
BY ..... DATE 4/3/95

Disclaimer Notice

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government or any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Superconducting Super Collider Laboratory is an equal opportunity employer.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SSCL-Preprint-542

Physics Research Integrated Development Environment (PRIDE)*

J. Burton and L. Cormell

Superconducting Super Collider Laboratory†
2550 Beckleymeade Ave.
Dallas, TX 75237, USA

December 1993

*Presented at the Third International Workshop on Software Engineering Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics.

†Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

PHYSICS RESEARCH INTEGRATED DEVELOPMENT ENVIRONMENT (PRIDE)

Jackie Burton, Larry Cormell
Superconducting Super Collider
2550 Beckleymeade Avenue
Dallas, TX 75237-3997

ABSTRACT

Past efforts to implement a Software Engineering approach to High Energy Physics computing have been met with significant resistance and have been, in many cases, only marginally successful. At least a portion of the problem has been the lack of an integrated development environment, tailored to High Energy Physics and incorporating a suite of Computer Aided Software Engineering tools. The Superconducting Super Collider Physics Research Division Computing Department is implementing pilot projects to develop just such an environment.

1. Introduction

Software Engineering (SE) technology continues to evolve in the direction of a coherent, integrated environment based on the complementary principles of Object Orientation (OO) and the Open Systems Environment (OSE). The unique teaming of physicists from both the Solenoidal Detector Collaboration (SDC) and the Gammas Electrons and Muons (GEM) collaboration with the Physicists and Software Engineers of the Physics Research Division (PRD) Computing Department (PRCD) at the Superconducting Super Collider Laboratory (SSCL) to develop such an environment, the Physics Research integrated development environment (PRide), portends a remarkable new phase in software development for High Energy Physics (HEP). It promises: 1) improved productivity in the development of software; 2) greater accuracy and reliability of the software; and 3) quicker turn-around time for both the results of analysis and for the implementation of new software tools/versions.

2. An Historical Perspective

Previous efforts to introduce structured SE techniques into the realm of HEP have met with, at best, mixed results. The adoption of formal SE methods by experimental high energy physicists has proceeded very slowly. Several recent articles by Loken[1], Knobloch[2], and Nash[3] provide interesting discussions of this lack of formal SE in HEP. It does seem strange that scientists in a field that constantly pushes the envelope by demanding the state of the art in cryogenics, electro-optics, structural materials, electronics and certainly computing hardware would not demand the same of SE. There are a number of historical, philosophical, and even

emotional reasons for this bias against, or at least reluctance to SE methods. While a discussion of this reluctance would probably be an excellent topic for a panel discussion, it is beyond the scope of this paper. We simply note in passing that structured analysis/structured design (SA/SD) methodologies were introduced in the early '70s and have been used successfully throughout the software industry since that time to reduce development time and costs and to improve product quality.

Despite this apparent bias against SE, several HEP experiments have recently made use of SE methodologies. The most notable are D0 and ALEPH. Both of these experiments have had considerable benefit from their use of SE methods and tools. The software developers in these experiments basically followed the classical life cycle or waterfall development plan[4] employed SA/SD formalism[5], and embodied their design in data flow diagrams (DFD) ala DeMarco[6].

The D0 collaboration used SA/SD methodology in the initial design and implementation of their software. DFDs and their associated data dictionaries were the primary tools used to develop the D0 software models. These along with structure charts and state transition diagrams provide the basic design, description and documentation of the D0 software system[7]. At the time that D0 began developing their software they were not able to obtain adequate computer-aided software engineering (CASE) tools to maintain their design and diagrams.

ALEPH has made extensive and effective use of DFDs throughout the life cycle of their software development effort[8]. In addition the ALEPH collaboration has introduced data modeling and data base techniques to provide organization for their analysis tasks. ADAMO (Aleph DAta MOdel) was developed by the Aleph collaboration where it is used in several programs including detector description, data acquisition, event reconstruction and display, data analysis and group administration[9]. ADAMO includes the elements of an entity-relationship (ER) model in a form suitable for HEP computing. Entity-relationship diagrams were drawn to provide graphical descriptions of the data structures and the relationships among them. Data dictionaries as derived from the DFDs were generated and maintained. A CASE tool from Tektronix was used to automate the process of drawing and updating the diagrams.

The ALEPH collaboration found several advantages to using SA/SD methods and CASE tools:

- There is a common/uniform style of work throughout the development effort.
- The extended analysis phase had a very positive aspect because it led to the clarification, documentation and easy communication of concepts and ideas to other colleagues.
- The design documentation was available before coding began, providing a powerful means to describe abstract details.
- The top level breakdown and organization of the design permitted partitioning of the work among several independent teams.
- Frequent walk-throughs and reviews for verification and validation provided increased quality assurance in the final products.

Thus far, those high energy physicists who have used SE methods and CASE tools have primarily followed waterfall development efforts with some rapid proto-typing and feedback (modified waterfall). The waterfall method can be quite effective, especially for fairly well defined, closed-end applications. However, for many problems, such as those typically encountered in HEP, the solutions can not been written in closed form at the beginning of the project. The input requirements and desired output may change during the development cycle. In these cases, the formalism of the waterfall method may be too rigid. Newer methods have attempted to address the need for dynamic flexibility within the development effort. Several of these include Dearnley and Mayhew's model[10], Boehm's spiral development model[11], and other variants of iterative prototyping methods[12 13 14 15]. There is no single silver bullet[16]. Each software project is unique with its own set of user requirements, desired level of quality assurance, available resources, deliverables and schedules. The development approach, planning, and design methodology must be tailored accordingly. In the end there is still no substitute for careful planning, lots of thinking, and plenty of hard work.

3. On-Going Efforts

Despite the limited success of previous efforts, there is still an awareness of the need for a better way to develop software. Numerous efforts are under way to incorporate sound software engineering techniques into HEP software development efforts in order to take advantage of rapidly changing hardware and software technologies. Several papers were presented at this conference last year on this topic including: 1) a proposal by F. Bruyant to design a coherent software framework for future HEP experiments; 2) a report by P. Kunz on SLAC B efforts to utilize Object Oriented Programming (OOP), C, C++ and UNIX; 3) a report by Durr and Lourens on a formal specification language for Object Oriented Design (OOD); 4) a report by A. Nilsson on an event generation toolkit in C++; 5) a report by C. Arnault on the use of OOA for the Delphi Online Event Display; and 6) a report by J. D. Shiers on HEPDB[17]. There are additional examples, all pointing to the conclusion that HEP software development is moving inexorably to an integrated, software engineering approach to solving the problems of HEP data capture, storage, filtering, reconstruction and analysis.

4. The SSCL Computing Environment

The SSCL PRCD, SDC and the GEM collaboration are in a unique position to take advantage of the leap in computer engineering efficiency and potential for improvement in software development cost-to-benefit ratios which presents itself with the implementation of an integrated software engineering development environment based on OO and OSE concepts. The lack of legacy constraints of outdated hardware, uncontrolled software and immovable bureaucracy provides the opportunity to plan and implement the most cost effective, scientifically feasible solution for the long term. Cost, resource and schedule constraints add to the attractiveness of alternative solutions

The viability of the utilization of scalable, loosely coupled networks of workstations as proven by the SSCL Physics Detector Simulation Facility (PDSF) argues successfully for the implementation of an open systems concept[18]. The need for the integration of a variety of analysis tools developed and used by physicists around the world with the latest commercial off

the shelf (COTS) SE tools points out the need for an applications programmer interface (API) within some integrated environment.

In the past, physicists have been able to rely on "free" manpower resources in the form of post-docs and graduate students. The scale and longevity of the efforts at the SSCL limit the effectiveness of such resources. The limited budget and resources allocated to PRCD and the collaborations for computer and software engineering makes it an imperative that alternatives be identified. The extensive requirements by both of the collaborations coupled with these limited resources begs for the implementation of reusable code and the sharing of resources to the maximum extent possible without endangering the integrity of either of the collaboration's experiments.

The long time scale for completion of the SSCL as well as the anticipated duration of each of the detector experiments requires the ability to provide technology forecasts beyond reasonable expectations of accuracy. As such, the implementation of interim software solutions must maximize the ease of modification, expansion and seamless replacement of the components of the environment. That is, software tools or sets of tools should be readily replaceable by newer versions or even new tools with minimum impact on the end user, the physicist.

5. The Beginnings Of PRide

The unique environment at the SSCL set the stage for an unprecedented combination of physics collaborations and laboratory computing department resources. Recognizing the need for an innovative solution to the problems of insufficient funding, limited personnel resources and tight schedule constraints, PRCD has long championed a confederated approach to maximizing commonalty in the solutions to the computer engineering and software development problems facing the collaborations[18]. The SDC Computing subgroup took the first steps towards such an environment by inviting members of the GEM collaboration computing subgroup to an SDC computing subgroup workshop at Fermi National Accelerator Laboratory (FNAL). Attendees left with a conviction that there was some common ground.

In May, 1993 an SDC computing workshop hosted by Rice University[19] featured full participation by both GEM collaboration computing subgroup members and members of PRCD. An effort was made, at that time, to establish and prioritize software development requirements shared by the two collaborations. Among those prioritized requirements were a requirement for a software framework environment and for an integrated Computer Aided Software Engineering (CASE) environment.

Subsequently, the first joint computing subgroup workshop was held at the SSCL[20]. At that time, a brief definition of each of the proposed joint projects was provided by the PRCD Computer Engineering Group. In addition, a detailed explanation of CASE and the integrated software engineering environment was presented. The presentation included a brief proposal to develop the PRide. More importantly, PRCD, the SDC and the GEM computing subgroups agreed to develop a Memorandum of Understanding for management of joint projects between PRCD and the two collaborations[21]. The Experimental Computing Management Committee (ECMC) was established to oversee initiation, assignment prioritization and allocation of resources to projects. Alternative approaches for the prioritization and resource allocation process were discussed and initial drafts were developed for the procedures by which joint projects would be initiated, approved, prioritized, managed and have resources assigned from PRCD and the collaborations[22 23 24]. In addition to the need to identify projects, priorities and

how resources were to be shared, the intent of this effort was to prevent the development of a client server relationship in which PRCD received requirements from the collaborations and returned with a complete system some time later that might or might not fulfill those requirements.

6. Framework Software

Among the first projects initiated by the ECMC was an effort to develop a standardized environment for joint off-line computing applications software to be known as the FRAMEWORK software. The concept of the FRAMEWORK software had grown out of the previous SDC core software efforts[25]. The FRAMEWORK software was seen as necessary to provide support to the various tools, utilities, application programs and subroutines that will be developed by the two collaborations.

7. PRide Reference Model Selection

Integral to the ECMC FRAMEWORK software development project was the requirement to develop an integrated software development environment, the PRide. In fact, the FRAMEWORK software can be seen as a component of PRide itself. It should be noted that the purpose of the reference model is to describe environments that support projects that engineer, develop or maintain computer based systems. It is explicitly aimed at establishing a conceptual basis for an environment, not at establishing any particular environment product. There is no part of the reference model that involves choosing a standard toolset. That is the purpose of the PRide development effort.

Evaluations of various reference models for integrated environments, including the Portable Common Tools Environment (PCTE)[26], the Reference Model for Frameworks of Software Engineering Environments[27], and the POSIX Open Systems Environments. POSIX model seeks to define the open systems environment. The NIST/ECMA reference model domain is the PSE frameworks that support software engineering.

These reviews led to the evaluation selection of the Next Generation Computer Resources (NGCR) program Project Support Environment (PSE) reference model for the initial attempts to define and model the PRide. The domain of the NGCR model encompasses each of the other standards and more in an attempt to establish a conceptual base for an environment.

8. NGCR PSE

The NGCR PSE reflects the aspects of a large collection of existing environment efforts and models. These include the Software Technology for Adaptable and Reliable Systems (STARS) program, the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environments (ISEE) working group, the European Computer Manufacturers Association (ECMA) TC33 Task Group on the Reference Model, the United States Air Force Software Life Cycle Support Environment (SLCSE) project, Honeywell's Engineering Information System (EIS) program, the Conceptual Environment Architecture Reference Model (CEARM) effort, and the standardization committees within the Institute for Electrical and Electronics Engineers (IEEE) and American National Standards Institute (ANSI) for POSIX and for CASE Tool Integration Models (CTIM)[28].

The approach of the model is comparable to the POSIX OSE and the NIST/ECMA Reference Model for Frameworks of Software Engineering Environments. The model incorporates the two different domains of interest of these environments: 1) the Open Systems Environment of Posix; and 2) the NIST/ECMA PSE frameworks that support software engineering.

Since both of these efforts developed independently, there are some minor discrepancies between them that are being resolved within the two affected agencies[29]. For example, the Network services described in POSIX are included in the Object Management Services Distribution and Location service and in the replication and synchronization service. Additional areas of interest are incorporated as well. For instance, it has been proposed that the NIST/ECMA Framework's Presentation service be revised to accommodate some of the emerging technologies in the area of Audio and Video Processing Service.

9. Case Tools

Before we look at the PRide, it is worthwhile to look at what CASE is in order to relate it to the PRide. We will discuss CASE in terms of the standard industry classification of Upper, Lower and ICASE or Integrated CASE. Upper CASE tools are those which support the initial phases of a software development project, to include systems analysis and trade-off studies. Upper CASE tools are generally selected to support the software development methodology selected by the organization such as SA/SD or one of the varieties of Object Oriented (OO) Analysis (OOA) and Design (OOD) such as Shlaer/Mellor[30] or Rumbaugh[31]. Lower CASE traditionally includes more programming specific tools such as editors, compilers, linker/loader, debuggers, restructures and other programming aides. ICASE tries to integrate all of the CASE tools into a coherent whole.

Examples of CASE tools may include products as simple as code editors like TECO, Vi, eve ,emacs, xedit, etc., debuggers or syntax checkers such as dbx, gdb or Kadb, or life cycle support products such as Cadre Technology's Teamwork or Interactive Development Environment's Software through Pictures which may provide some automated project management, analysis and design, quality assurance, configuration management and testing tools as well as code generation support. Generally speaking, CASE tools provide automated support for some or all of the following activities:

- REQUIREMENTS MODELING USING DIAGRAMS - CASE tools in this area generally provide for initial graphic representation of the system, subsequent changes to and output of in-progress or completed diagrams. Object input validation and diagram manipulation rules are provided. It should be noted that graphic representations have not yet become standardized within the industry.

- MODEL VALIDATION - Tools in this area check for model completeness, and identify inconsistencies or errors. Validation may be interactive, batch or both.

- SPECIFICATION DEVELOPMENT - Tools in this category provide for the conversion of the initial model into preliminary and detailed design specifications.

- DESIGN VALIDATION - Tools in this area check for design completeness, and identify inconsistencies or errors. At this time, most such tools are platform dependent.

- CODE GENERATION - Tools in this category support the transformation of detailed design specifications into executable code. Again, code generators are typically specific to a single platform.

- TESTING - Automated testing support includes both the automated generation of test plans and procedures and the exercising of the target system either directly or through the use of a simulation of the target environment.

- PROJECT MANAGEMENT - These tools provide a framework for the recording of the work planned, the actual work done, and an assessment of the work to be completed.

- DOCUMENTATION PRODUCTION - These tools provide for the automated generation of documentation, in a format based on site specific templates, from information provided by the specification models for both the requirements and the design.

- RE-ENGINEERING OF EXISTING APPLICATIONS - These tools provide support for the automated documentation of existing code, the analysis of existing systems and the subsequent redesign/optimization of the existing systems either manually or automatically[32].

10. Description Of The Pride Model

In the development of the PRide, we have chosen a course which seeks to optimize the advantages of CASE tool utilization by integrating them into a single, cohesive environment. Among the advantages identified for this type of integrated environment are:

- The maintenance of a single set of object definitions, particularly data definitions, in a variety of development efforts.

- The ability to maintain project continuity via the access of the same information from all phases in the development process.

- The ability to identify targets for re-use and to re-use objects in different application development projects.

- The ability to integrate project management, analysis, design, and development efforts to maximize the opportunities for the use of automated tools and to prevent the disconnects prevalent in other development environments which result in incorrect or missed requirements in the target implementation[33].

The PRide will provide services which will span the functionality of a populated environment. We define an environment to be a collection of software and hardware components, with a degree of compatibility sufficient to render these components harmonious.

Services are defined as an environment's capabilities. Those capabilities which directly support an end-user are referred to as tools. The components that comprise an actual infrastructure are referred to as the framework.

The Pride services will be grouped in different categories reflecting degrees of abstraction, granularity, or functionality. The highest level division classifies services either as end-user or framework services. End user services will be further subdivided into Technical Engineering, Technical Management, Project Management, and Support Services. The first three of these groups partition the execution of a project into engineering, management, and engineering management. The fourth group , Support Services, is, per the NGCR reference model, orthogonal to the other three, since it includes capabilities potentially used by all other users[34]. The logical relationship of these service groups is illustrated in Figure 1. Framework services form a central core with a potential relationship to all other services in the environment. Support services underlie all the other end-user services. The remaining groups provide an envelope around the framework services and utilize the support services. The boundaries do not explicitly indicate interfaces, as the representations are at the service groups level, not the services level. To emphasize this point, the service groups might be represented as in Figure 2.

Figure 1

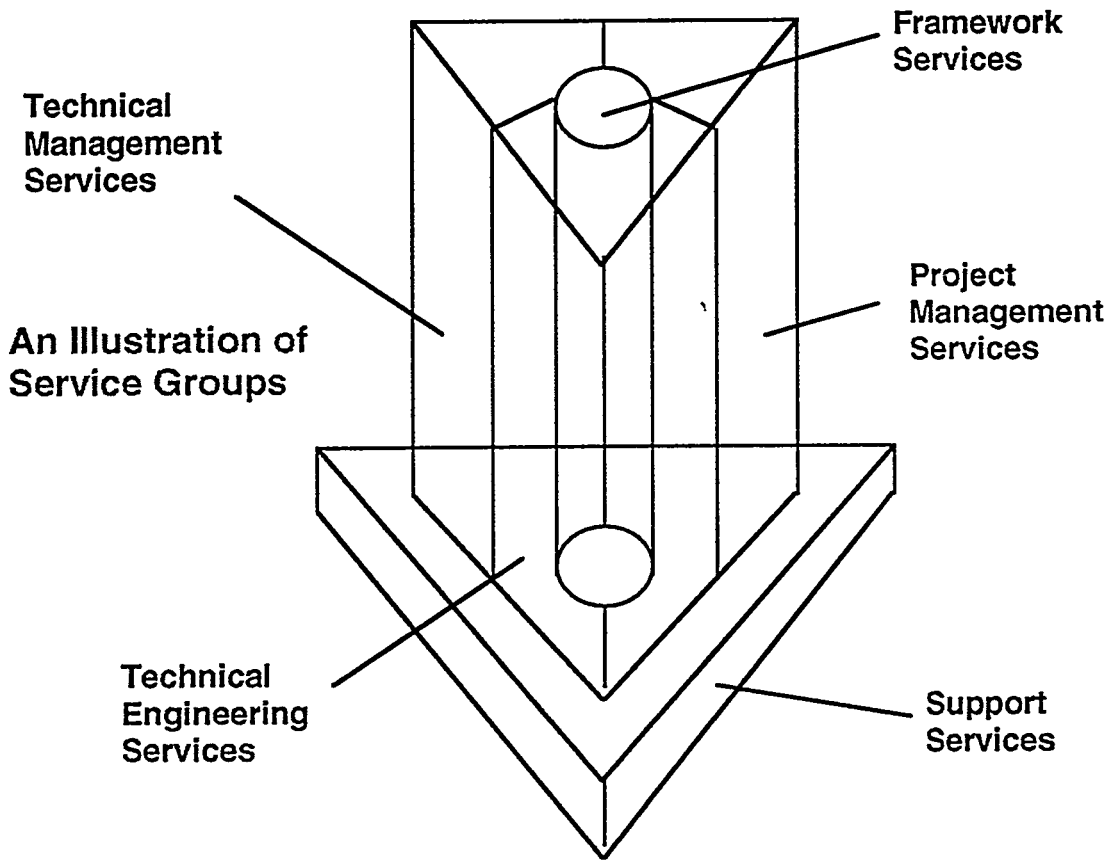
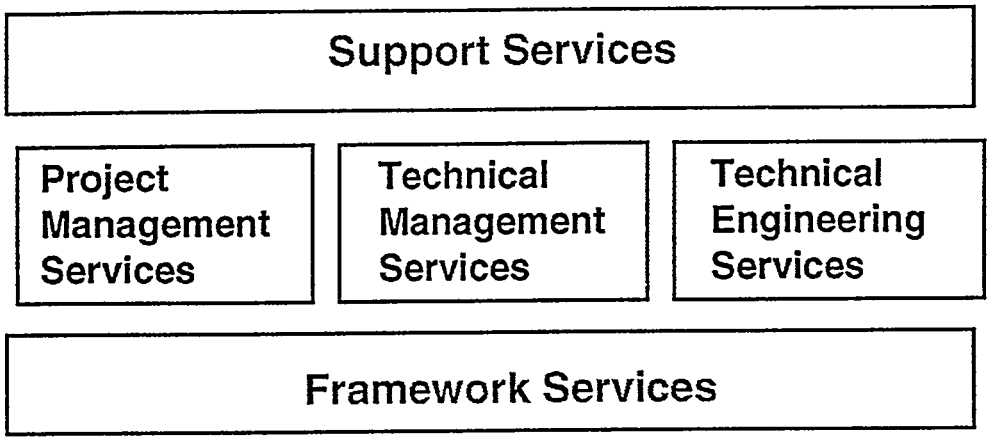


Figure 2



Another Illustration of Service Groups

Each of the end user service categories is further subdivided by engineering domain, user role, or life cycle phase. Technical engineering services are directed toward the technical aspects of project development. These services are organized by specific engineering domain. Tasks typically include designing and coding which require services such as compilation and testing. Technical management provides services that are closely related to engineering activities; these include services which provide a managerial complement to engineering activities in the areas of configuration management, reuse, and metrics. Project management services are relevant to the overall success of the SSCL. They include such things as scheduling, planning, and tracking overall progress of a project. Support services focus on tasks and activities common among all users of the PRide. They include a group of common services for information processing, as well as publishing, user communication, presentation, and administration services.

The framework service categories include Operating System, Object Management, Process Management, Policy Enforcement, User Interface, Communication, Network, and User Command Interface services.

Real software components span various service groups with many components considered to be end-user tools also providing capabilities properly regarded by the NGCR reference model as framework services. The model provides a common conceptual basis against which to examine these environmental implementations. The boundary between service groups, particularly the boundary between end-user and framework services is a dynamic one that changes over time.

11. Development Of PRide

The development of PRide is following an iterative multi-stage development path. Parallel efforts are underway at the SSCL to: 1) identify the current physics computing environment both within PRCD and the collaborations; 2) to establish an approach to developing framework services within the ECMC FRAMEWORK project[35]; and 3) to acquire, build and integrate components of the PRide through innovative acquisition techniques, including the utilization of

Creative Research and Development Agreements (CRADAs) which would allow software developed by teams of PRCD engineers and commercial vendors to be made available at little or no cost throughout the collaborations.

12. References

1. S. C. Loken, Software Engineering: What Do Experiments Need?, Proceedings of the International Conference on Computing in High Energy Physics '92 (1992) 87.
2. J. Knobloch, Reality of Software Engineering in High Energy Physics, Proceedings of the International Conference on Computing in High Energy Physics '91 (1991) 291.
3. T. Nash, High Energy Physics Experiment Triggers and the Trustworthiness of Software, Fermilab-CONF-91/270, August, 1991.
4. W. W. Royce, Managing the Development of Large Software Systems: Concepts and Techniques, in the Proceedings of WESCON (1970)
5. B. W. Boehm, Software Engineering, IEEE Transactions on Computing (1976) 1226.
6. M. Paige-Jones, The Practical Guide to Structured Systems Design, Yourdon Press (1980).
7. T. DeMarco, Structured Analysis and System Specification, Yourdon Press (1978).
8. J. Linneman, et al., The Use of SA/SD Methods in D0 Software, Computer Physics Communications 45 (1987) 245.
9. G. Kellner, Development of Software for ALEPH Using Structured Techniques, Computer Physics Communications 45 (1987) 229.
10. S. M. Fisher and P. Palazzi, Using a Data Model from Software Design to Data Analysis: What Have We Learned? , Computer Physics Communications 57 (1989) 169.
11. P. A. Dearnley and P. J. Mayhew, In Favour of System Prototypes and Their Integration into the Systems Development Cycle, The Computer Journal 26 (1) (1983) 36.
12. B. W. Boehm, IEEE Computer, 21 (1988) 5.
13. B. Boehm, et al., Prototyping Versus Specifying: A Multiproject Experiment, IEEE Transactions on Software Engineering SE-10 (1984) 290.
14. F.P. Brooks, No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, 20 (4) April 1987.
15. K. S. Lantz, The Prototyping Methodology, Prentice-Hall (1986)
16. J. L. Connell and L. B. Shafer, Structured Rapid Prototyping: An Evolutionary Approach to Software Development, Yourdon Press (1989).
17. In the Proceedings of the Second International Workshop on Software Engineering, Artificial Intelligence and Neural Networks in High Energy Physics.
18. B. Scipioni, et.al., Physics Detector Simulation Facility Phase II System Software Description, Proceedings of the Conference on Computing in HEP, Annecy (1992) 597.

19. L. Cormell, Experimental Systems Computing, SSCL-N-814, February, 1993.

20. I. Gaines ed., Proceedings, SDC Computing Workshop, Rice University, May 17-19, 1993.

21. I. Gaines ed.l Proceedings, Joint GEM/SDC/PRCD Computing Workshop, Superconducting Super Collider Lbaoratory, July 1993.

22. Memorandum Of Understanding, R40-000006

23. Project Initiation, R40-000007

24. SSCL PRCD PRACTICE #R40-000008 Project Assignment Plan.

25. SSCL PRCD PRACTICE #R40-00009Project Execution Plan.

26. S. Frederiksen, I. Gaines, L. Price editors, SDC Off-Line Software Concepts, version 1, December 5, 1992.

27. ECMA, Technical Report TR/55, European Computer Manufacturer's Association December 1991.

28. NIST, Reference Model for Frameworks of Software Engineering Environments, NIST Special Publication 500-201, December, 1991.

29. PSESWG, Reference Model for Project Support Environments, Next Generation Computer Resources Program, Project Support Environment Standards Working Group, United States Navy, March 1, 1993, pp2.

30. S. Shlaer, S.J. Mellor, Object Oriented Systems Analysis, Modeling the World in Data, Yourdon PRess Computing Series, Prentice-Hall, 1988.

31. J. Rumbaugh et.al., Object Oriented Modeling and Design, Prentice-Hall, 1991.

32. PSESWG, Reference Model for Project Support Environments, Next Generation Computer Resources Program, Project Support Environment Standards Working Group, United States Navy, March 1, 1993, pp2.

33. John Parkinson, Making CASE Work, NCC Blackwell Limited, 1991, pp37-38.

34. ibid pg39.

35. PSESWG, Reference Model for Project Support Environments, Next Generation Computer Resources Program, Project Support Environment Standards Working Group, United States Navy, March 1, 1993, pp7-8.

36. Frederiksen, et.al., FRAMEWORK Software Project Assignment Plan, SSCL, September 1993.