

Software Tools, Languages and Environments: a Summary

Ruth. Pordes,

*Fermi National Accelerator Laboratory, PO Box 500, Batavia, IL
60510, Tel: 708-840-3921*

The importance of methods, tools and infrastructure to develop and maintain HEP and Nuclear Physics software is increasing with the size and complexity of these systems. This paper summarizes talks presented at CHEP 1995 in the sessions on these issues.

1 Introduction

The session covered software tools and engineering methods that help experiment collaborations and experimental science institutions develop the software systems they need for the future. Other presentations at the conference were convincing in their discussion of the increase in the magnitude and complexity of the software generation, maintenance and upgrade tasks ahead. The long time frames of future experiments further increase the emphasis on planning and management needed to provide for the evolution and support of the software. Some twenty five papers were presented covering:

- Software Tools and Environments - software that aids the software application developer, user and maintainer in their task.
- Languages- evaluation of a particular programming language; integration of systems of many languages and computer platforms.
- Object Oriented Technologies - CASE tools for the application of OO design and development methods. Software tools that aid the software development managers as well as the developers.
- Software tools for the configuration and use of computer systems.

Additionally the papers in this session concentrate on the essential “glue” that must be present for successful software system design and construction and maintenance. This is especially important when the programs are developed by many different people.

Eight presentations covered work in Object Oriented Analysis and Design concentrated in groups preparing for BaBar - under time constraints - and the “long haul” towards the LHC. Work continues in data acquisition control in existing and planned experiments. Mixed language programming rather than single programming language wars was the issue. A couple of presentations showed work in progress on software metrics and quality control - essential work on metrics and process for delivery and maintenance of software over a long period.

The requirement to provide transparencies online before the conference added to the knowledge gained and greatly helped those participating over long distance (and in this case the summarizer!). The excellent help from the “desk” in copying transparencies and papers is gratefully acknowledged.

The plenary sessions provided a well tailored complement to the parallel session talks.

* This work is sponsored by DOE contract No. DE-AC02-76CH03000

The presentations from outside the HEP and Nuclear Physics fields were stimulating and thought provoking. While the plenary talks are not covered in detail here, the contributions of such talks to the conference were invaluable.

2 Application Development and System Environments

Institutions with a large user community are continuing their efforts towards consistency in “user environment look and feel”. They aim to provide software layers above the operating system but below the application to make the average application developer’s life easier - both programmer’s and physicist’s. From the talks presented it seems that we are still incapable of coming to consensus across the Atlantic. This issue will become more of a problem as developments for LHC experiments gets underway. We can only hope that HEPiX will continue and collaboration and “trust” will increase.

CERN presented their Common Unix Environment - based on a common hardware architecture and standard “naming” tools. Within HEPiX a CERN, Desy collaboration have come to good agreement on the shell and “. ” files, and are starting to implement a common X environment for the average user. Fermilab’s Unix Environment (FUE) - one of the first into the fray - has the expanded goal of providing tools and infrastructure for development of software systems. FUE is also looking towards the future of multi-cultures more diverse than UNIX. The ADAM project at CERN provides a “PC” lookalike desktop for launching applications on a remote UNIX host.

It is clear that system managers have to provide a comfortable environment for both the initiates who are coming off VMS and VM, and for the experts who delight in using all possible arcane UNIX functions they can. Additionally, they need to prepare quickly for life where some consistency across PC Operating Systems and Unix is a goal

Several significant issues remain to be addressed by the community:

- Interdependencies and versions that make up an integrated system (which were not addressed at all by talk on Components and OLE)
- Run time vs. compile time configuration issues
- How to accommodate the range in sophistication and needs of users - young sophisticates to old fogies

3 Languages

The emphasis of the talks presented was how to live in multi-lingual harmony. C++ is accepted as a serious language “to be used”. This was a change from CHEP ‘94 where the issue was “is C++ a possibility”. Paul Kunz continued to encourage us to train and educate ourselves as we start the serious move from Fortran to C++. The overall impression given was that the HEP community is starting the migration in earnest. However, Viktor Decyk and Charles Norton from the Numerical Tokamak Project showed that for their very large numerical computations recoding in C++ gave no advantage in efficiency. This provoked expressions of dissent from the audience. Their presentation, as well as that from Digital on the ease of designing and implementing parallel programs in Fortran ‘90 on a DEC

MPF system, left the idea that Fortran '90 should not be overlooked if speed and efficiency are important criteria. Moose was the only project to report that it uses Eiffel.

Attention was given to mixing languages in a single application, Zeus offer “wrapper” files to allow easier integration of Fortran and C libraries. However, the more complex issues of structure translation are not yet addressed.

IDL is being adopted as a higher level specification language that to allow language independent implementation. It allows for the implementation of a single design to be in multiple languages, with integration of different languages on different computers occurring in a seamless way. This showed their youth by being accompanied by issues of consistency of compilers, interfaces to other languages, support from CASE tools etc. Dave Quarrie, reporting on evaluations he is doing for BaBar gave concrete examples of how this language independent approach could be accomplished in practice.

Mixed operating system programming clearly needs as much attention as mixed language programming. We need to pay attention to the conceptual problems of the every day world of mixed languages and systems; and must be prepared to learn, if not converse, in more than one. The knowledge of a language needed to read and interpret (debug? interface to?) is at a very different level to the fluency needed to write good code and algorithms. Given the requirements of processing and analysing physics data, we still cannot afford to compromise the quality of the code we write for the ability to write in a chosen language. Better to define good interfaces and write the best code we can in the language of our choice.

This topic was complemented by the talk from David Weiss of AT&T on FAST- a systematic process for generating domain specific languages for implementation of members of program families. Families are the results of analysing software systems as classes of common software organization and groupings - reimplementations of the same overall concepts.

4 Object Oriented Design Methodologies and Supporting Tools

Experiments starting software development now are using formal OO design methodologies and supporting CASE tools for design and implementation. The table below illustrates the diversity of those actually being used. The RD13/CMS data acquisition project, using OMW, and the ATLAS trigger reconstruction group showed the most concrete progress - with running applications whose implementation was generated directly from the design captured by the CASE tool.

RD13 used the OMW case tool, the Kappa programming environment, and the Itasca Object Oriented Data base system, to implement parts of a data acquisition system. They have run into the expected problems of platform support (no support for real time OS) and integrating the implementations from these encapsulated environments.

As was true for Structured Analysis and Design the final worth of the concepts and effort put into the methods, will not be known until a particular project has been underway for several years. I look towards CHEP 1997 as an opportunity to invite these speakers to report on their projects “2 Years Later”.

Table 1: Use of OO CASE Tools.

	Goal	Method	Tool
BaBar	Data acquisition development		MarkV?
PHENIX	Design higher levels of online system	Shlaer Mellor	ObjectTeam
RD13	Design of data acquisition	OOIE	OMW
ATLAS (RD41)	Electron track reconstruction	Class Relationship	Objectteering
CMS (RD41)	Reconstruction	Booch	ROSE C++
Katayama	R&D on HEP code	Pattern analysis	

The presentations showed that the traditional problems remain. The rigidity of the specific tool used affects the success of the developments. So does the willingness of the developers to commit to the tool for the long term. The problem of good integration of use of the tool with the rest of the software delivery process has not yet been solved. The commercial world of CASE tools is still fluid and the longevity of any particular one is always a question. The attention span of the developers and their management is still an open issue. To retrofit the design to the actual implementation is necessary for long term maintenance and support - but remains difficult. However, use of the OO methodologies in the field is now ubiquitous.

Pattern analysis encourages the coder to think in a more abstract way about the small scale patterns in the implemented code. The code is then crafted to provide an elegant solution at a local level. While the goal is to encourage reuse and reduce maintenance of the software algorithms, this approach does not replace the need for system design and analysis. Pattern Analysis as a software engineering method is starting to gain acceptance. Dr. Katayama gave a very worthwhile talk to introduce us to the topic and the potential for its applicability to our needs.

5 Distributed Processing and Control:

Aleph and Zeus, two long running experiments, reported on upgrades to their distributed run control systems to address bottlenecks in the systems and address the longer term needs for maintainability and user satisfaction. DART reported on attempts to prevent such bottlenecks by designing for multicast services.

New distributed control systems, currently in the design phase, propose to use CORBA. A small prototype system from BaBar demonstrated that applications can be built using the standard. However, the paper from Dave Quarrie detailed that IDL compilers are a long way from being standard, that Fortran interfacing is an R&D project for a high energy physicist, and there is significant extra code generated for an application and

throughput measurements must be made. More, larger implementations using the standard are needed. There remains the concern, also, that Microsoft's OLE will overtake Corba as it moves into the main arena.

6 User Interfaces

There were several reports of projects whose user interface is based on tcl/tk. The Adam project is using the OO variant - Incr-tcl. There was no discussion of common tools for "what comes next" for analysis or control programs. There was no discussion of the "command line interface" or "GUI" battles. 3-D user interface and control is still at the stage of R&D.

7 Software Quality Control and Analysis

The software industry talks a lot about software quality standards and metrics for measuring them. However, there were few talks on the subject at CHEP95. Our traditional approach has been to say that the process and bureaucracy imposed on commercial enterprise will be detrimental to the experimental and creative nature of scientific research. However, with boxes such as "Software Manager" now appearing at the high levels of HEP collaboration organization charts, it is clear that software is becoming recognized as of major importance and concern. Two talks at the conference reported on concrete attempts in this area:

With what was claimed to be 2 FTE-weeks of work, Eric Lancon of Saclay analysed 116K lines of Aleph Fortran analysis code. He used Logiscope to measure and compare the "quality" of 1990 and 1995 versions of the code. His results confirmed, what one is tended to guess at, that the "worst quality routines" required the most modification during the five years of the measured lifetime. This would indicate that improving the measured quality at the time of initial coding will save maintenance costs in the longer term.

At Fermilab, automated scripts for checking of the collaboration software standards and methods have been developed and are invoked whenever the analysis software is built. These scripts must also take account of the fact that too much information and trivial complaints are counterproductive.

While the details of each project were interesting the overall approach - which accepts the worth of putting manpower into developing and applying such tools - also bears consideration. Both projects provided non-emotional and impersonal checking of the standards imposed. However, human factors must have prime influence on the way the checking and reporting are done in order to have an effective response.

8 Build, Buy, Share

Everywhere we see increased emphasis on buying, borrowing, and using software already developed and in use in the general market place. We need to identify those areas of the HEP software endeavour that can benefit from this the most. Projects to evaluate, com-

pare, and prototype applications using the different products will benefit the whole community (if the results are disseminated). In the talk from the Moose collaboration we were challenged: “to give access to increasingly sophisticated software while hiding its complexity”. When taking software developed by others we must also solve “giving access to increasingly sophisticated software and debugging it when it has problems”

We should not forget that the software tools and applications created in HEP and Nuclear Physics can have utility in the wider community. At this conference, a small example of this was presented by Professor Vaz of LAFEX who used traditional HEP production control code to aid in the statistical analysis of electronic circuit design. The introduction of the Fermilab Software Tools program is an attempt by Fermilab to provide an umbrella infrastructure through which use by clients outside the laboratory can be supported.

There was little discussion of “shared software”. This is becoming increasingly interesting to development groups in accelerator and telescope controls but was not emphasized at this conference. I suspect this is something we will see gain prominence at CHEP 1997.

9 Conclusions

The overall impression is that real accomplishments still require a lot of mundane work and there is no miracle on the horizon. We clearly need management commitment to the adoption of a significant software tool or software engineering method. This requires persistence in managers, designers, coders and maintainers. In some ways this is more important than discussions of “persistent objects”.

Software systems and applications running across multiple operating systems are a must. Work in understanding the requirements, constraints and opportunities is needed. Projects to explore how to live and flourish within these environments are essential. We must be agile in moving to new systems and languages as they continue to arrive (at an ever increasing rate?) from the world outside.

We must continue to be aggressive in looking wider than the forces of our local field for ideas, applications and software tools. While the Big Blue and Soft Gentle Giants are promoting after world wide saturation for their products, we should not be scared to adopt solutions from our peers in other scientific disciplines, or to collaborate on the development of innovative tools targeted for our special needs.

Finally, we must not forget that without organizational and reward structure to encourage work on hard and “boring” problems in general they just don’t get done (or at least reported - which presumably should be one of the rewards)

10 References

This paper references many papers presented at the CHEP95 conference. Thanks are due to those who spent the time to write up their research and results for the benefit of the community at large. Readers are encouraged to read the papers referenced for more detail and information on the topics mentioned.