Summary of Session E, "Tools and Methods"

L.A.T.Bauerdick

DESY, Notkestr. 85, D–22603 Hamburg, Germany

The use of software engineering methods and tools is becoming mainstream in the efforts to develop the complex software systems for the HEP experiments now under construction. The paper summarizes the presentations on this subject given at CHEP97.

1 Introduction

The large number of submissions to the session on "Tools and Methods" proves the active interest in the subject. The conveners accepted 25 submitted papers for presentation in the parallel and poster sessions. In addition, the plenary program had a panel discussion on the transition to object oriented (OO) programming, and an invited talk on software engineering support.

The bulk of submissions which I will summarize in this paper can be categorized as presentations on

- OO software projects and project management,
- experience with OO analysis and design methods, tools and programming,
- engineering data management and information systems (EDMS).

In the HEP experiments now under construction the production of software receives more attention than ever. Preparation and planning for software starts in an early phase of the projects, reflecting that software is considered to be of vital importance. Also, software projects are now mostly treated as engineering projects. The expected functionality is carefully analyzed, requirements get documented early on and project management is an issue. Treating the project as an engineering subject, the use of engineering methods and tools is conceived necessary for successful completion.

Preprint submitted to Elsevier Preprint *8 July 1997*

2 OO Software Projects

HEP experiments now being constructed or upgraded at FNAL, SLAC, DESY and RHIC will enter their running phase on a time scale around the year 2000. Presentations were given by CDF, D0, BaBar, HERA-B, Phenix. All of them embark on using object oriented approaches, despite the severe time pressures and little previous experience.

The LHC experiments have a longer time scale until around 2005. The computing technical proposals of Atlas and CMS were submitted earlier this year. Both experiments will use OO programming and the C++ language. This decision to some extend has been influenced by OO prototyping work that was done e.g. by the Moose collaboration, which presented its final conclusions in the conference. Its members are by now dissolved in the computing group of the LHC experiments.

OO is now mainstream and its well known advantages have been identified to be applicable to the field of HEP. Some large OO software projects like e.g. Geant4 or the BaBar reconstruction are well underway, and expertise on using OO techniques has been acquired amongst a growing community of people.

Geant4 is the largest OO project in HEP, a worldwide collaboration of some 100 members. It does a re-implementation of the Geant simulation package, using OO design and programming in the C++ language. An alpha-version was timely released during the conference, proving that distributed software production involving application domain experts (i.e. physicists) and OO programming can be done successfully. It also shows the large interest amongst physicists to get involved and trained in modern software techniques.

With OO programming and C++ becoming mainstream, the need for support on physics software in the language is becoming obvious. The Fermilab Physics Class Library project at FNAL and the LHC++ project at CERN address the needs for class libraries and tools to provide the equivalent of the current CERNLIB, building upon earlier projects like CLHEP. LHC++ adopted the strategy to use commercially available components as much as possible, and to add HEP specific functionality. The outcome of these projects will show in what way C++ will redeem the promise of reusability of OO software.

3 Methods, Notation, and CASE Tools

The fact that different projects embark on different methodologies, like Shlaer-Mellor (e.g. Phenix), OMT (e.g. Moose–Atlas) or Booch (e.g. Geant4) emphasizes that not one method is fully satisfactory. The newly defined Unified Modeling Language (UML) is forthcoming.

Often a method is used rather as a notational device, not necessarily as a design and software engineering methodology. It provides a high level language for documentation, discussion of designs and implementations. Using a well defined notation allows to some extend the automatic translation back and forth between design documents in the "high level" graphical notation and the implementation in the "low level" programming language — if the CASE tools support it.

CASE tools are considered useful for the design especially of large systems. However, often they support just one method, are poorly customizable, weak in code generation, have bad user interfaces, and also they are considered to be quite expensive for what they offer. There is good use for tools that allow to translate existing C++ code back to diagrams, like SNIFF+ which is used with StP in Atlas DAQ. This helps to keep the documentation up to date and to communicate more easily the design of existing code.

The reverse process of automatic code generation from the model has in some cases also successfully been used. It gives the advantage of working at the level of a graphical model instead of the C++ source code. For the Phenix online system a formal method (Shlaer-Mellor OOA) was used to define the architecture of the run control and hardware configuration system. A commercially available code generator was employed to implement the system in C++. Code generation was also used in the Atlas DAQ area. The run controller finite state machine was defined in StP/OMT and the code to implement it was produced automatically.

4 Languages

The times of language wars are over. The struggle of Moose over the use of Eiffel versus C++ exemplifies the importance of staying mainstream, in order to get decent compiler and tool support. Today the language of choice is C++. However, there is significant interest in using Java, and many smaller projects have successfully used Java and the WWW as the programming platform.

Employing design methods requires acquaintance with its notations for Class Diagrams, State Transition Diagrams etc. These are also useful for communicating on designs and implementations using it as a high level graphical language.

5 The Software Process

In addition to the choice of technology, i.e. OO analysis, design and implementation methods, software engineering tools etc, also project management procedures receive an increasing interest.

Using a defined software development process aims at ensuring production of quality software, i.e. promoting conformance to requirements and ease of maintenance. Also, resource and schedule constraints have to be handled, and timely and appropriate corrective actions to maintain conformance to goals should be taken.

Some projects, most notably the LHC experiments, are setting up a scheme for the management of the development process, defining standards and procedures. The Atlas software process was presented, which defines deliverables, including requirement documents, design and implementation, a sequence of activities implementing an iterative cyclic approach, and also the roles and hierarchies of architects and developers.

To ensure adequate qualification of programmers, training is considered to be very important. Those advertising formal approaches to the software process, e.g. the Atlas representatives, also considered professional training graded to the different levels in the project hierarchy. In some projects external consultants have been successfully employed.

There was vivid discussion on how applicable and useful the formal definition of the software development process is, and how realistic it is that it will be followed in the real world. Also, there was not unanimous agreement that in HEP the managerial structures even exist as of yet to promote a formal approach to project management. It was however agreed upon that some definition of a software process is needed at least for the larger projects, and that most importantly the people involved must believe in its usefulness.

6 Engineering Support for Software Development

The HEP laboratories and groups depend on an infrastructure of support for mechanical and electronics engineering. For software engineering this structure is largely non-existent.

In her invited plenary talk G.Pawlitzek described her model for proving a successful software engineering support structure and service center at the DLR. Support on CASE tools and software engineering experience is brought to research groups from a software competence center, which provides professional and accountable services and ready to use solutions to the science and research groups.

In her list of "key success factors" of software projects she ranks the use of appropriate software engineering standards and methods subordinate to a realistic estimation of feasibility and effort, and working with a good and motivated team in a structured manner, having basic knowledge and understanding of the principles of software engineering.

7 Engineering Data Management and Information Systems

There is a considerable interest at the large labs for EDMS. Projects on data management and information systems were presented from CERN, DESY and FNAL addressing mechanical engineering, workflow management, software documentation and business data.

The goal is to integrate vast amounts of usually unstructured information of different kinds into one data management system. All data pertaining e.g. to an engineering project should be accessible online, facilitating searches in and navigation around documents. Emphasis is given on a WWW based user interface.

An EDMS contains all data and documents, allows work and workflow management, has version and validity control and configuration management. The data proper is stored in the data vault, and additional meta-data provides structure and semantics to the data.

The use of these tools is considered necessary to allow distributed engineering to becomes manageable. As an example the workflow management for the worldwide production of the thousands of crystals for the CMS electromagnetic calorimeter was presented.

EDMS and software engineering are closely related. EDMS data modeling uses software engineering techniques, like OO design and ODBMS. On the other hand software engineering can profit from having EDMS-like services for the documents pertaining to software projects. Indeed, the Light-OO project presented at this conference offers an information management system for software documentation and maintenance based on hypertext and the WWW.

8 Conclusions

Software engineering offers solutions to the challenges of building the complex systems for future HEP experiments. The OO paradigm is now mainstream, and software engineering methods and tools have been successful in large and distributed software projects. Management commitment to providing the necessary resources and support structures is needed.

Acknowledgments are due to the co-conveners of this session, Kors Bos of Nikhef and Ruth Pordes of FNAL, to the contributors to this session and to the organizers of the conference.