

Engineering a Shared Software Base Between High Energy Physics Experiments.

BaBar: Robert Jacobsen, Marc Turcotte for the BaBar Computing Group
CDF: Elizabeth Sexton-Kennedy
CLEO: Christopher D. Jones, Martin Lohner, Simon Patton [1]

[1] Now with the BaBar Collaboration.

We describe an ongoing infrastructure-level scientific software engineering collaboration between BaBar, CDF and CLEO. Focus is on sharing actual code and designs engineered from requirements identified by the respective experimental groups. The benefits from exhaustive testing prior to code mergers or design sharing, are evident in the products. High integrity code has resulted from the mergers. Significant gains were made by everyone as is evidenced in the form of contributed new functionality that limited individual manpower would not have otherwise allowed. Early objectives and expectations for this collaborative effort are therefore supported with successful implementations. The key elements of this successful collaborative strategy are discussed and examples of shared/reused code and shared/reused designs are given. Issues in code management and maintenance are also discussed. The impact of this multi-experiment collaboration is found to be positive.

1) INTRODUCTION

The software subsystems of high energy physics experiments amount to a significant fraction of the total effort in building the experiment. This is mostly in terms of manpower and, increasingly, in terms of commercial software products and maintenance costs. This paper describes an ongoing effort between BaBar, CDF and CLEO to abstract out common offline functionality and share both designs and code. The primary goal is to pool resources and foster innovation through collaboration. A definite byproduct is in the cost savings achieved.

2) FRAMEWORK: AN EXAMPLE OF ABSTRACTED FUNCTIONALITY

The product being shared is the experiment physics event reconstruction framework. We describe the Framework only in terms of its broad features and essential functionality. More details can be found in [Ref. 1]. In broad terms, the Framework is the physics event reconstruction organizing and enabling layer. It is that object-oriented set of classes and the way they collaborate that is common to all offline reconstruction code of HEP experiments. We identify several key aspects of the Framework and describe them in some details.

2.1) THE RECONSTRUCTION UNITS

This is the part in the Framework that interfaces to the client users. In BaBar and CDF, we use inheritance to expose a reconstruction interface to users. The smallest unit of reconstruction is called a module. Modules can be time-ordered into new objects called sequences. Sequences can be made of either modules and/or other sequences. The time-

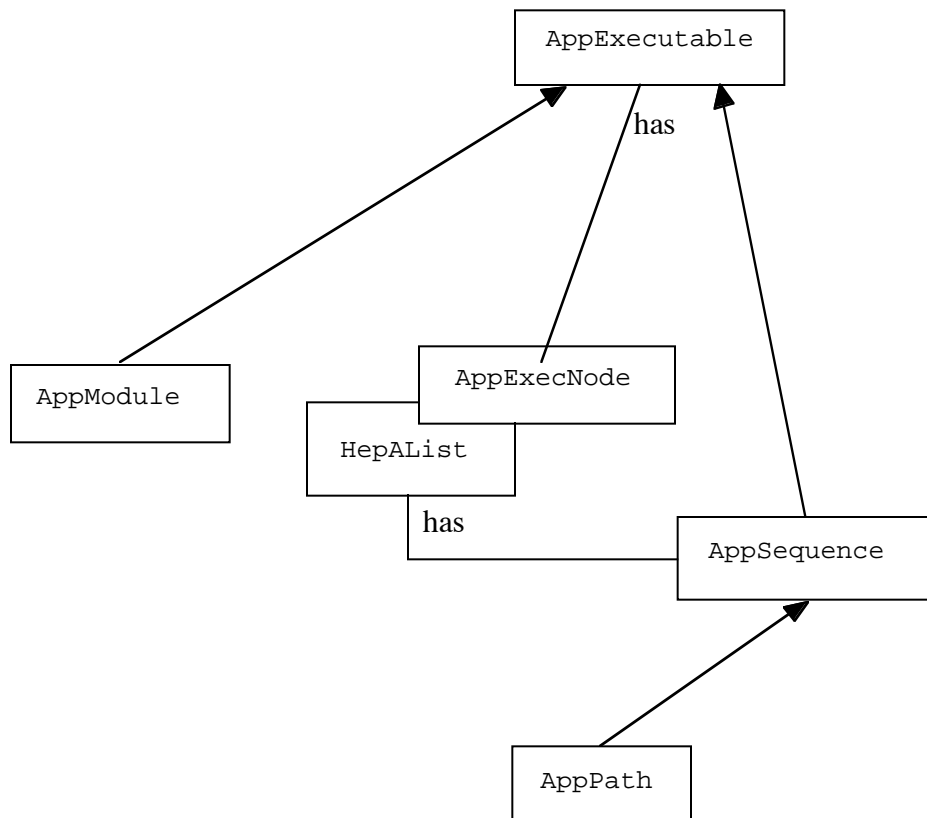


Figure 1: The object structure of reconstruction units in the BaBar Framework. The arrow indicates inheritance and the label "has" indicates containment.

ordered series of sequences and/or modules where an event is fed in at the beginning by a special purpose event input module is known as a path. Another kind of special purpose module, the event output module, can be appended to a path and takes care of outputting the events. In BaBar and CDF, we only allow sequential execution of paths. Paths are executed in sequence but may be terminated before completion by a filter module. Figure 1 shows the essentials of the object structure in the CDF/BaBar Framework.

2.2) THE EXECUTION SETUP

This is the part of the Framework that the users interact with to create and setup the sequences and paths. In BaBar and CDF, we use Tcl to interact with the user at the command line level and Tk graphically. New Tcl commands can be created by users of modules and added to the known list. This is done by encapsulation of the Tcl mechanisms to create new commands. A graphical interface is in early deployment phase.

2.3) THE EXECUTION ENGINE

The Execution Engine is that part of Framework that walks a high energy physics event object through the paths and therefore through all the execution units in a time-ordered way. In BaBar, this object is uncharacteristically known simply as the AppFramework but is also sometimes referred to informally as the Execution Web Manager. This execution engine insures that a given module instance is executed once per event even though it is part of many paths. This is a design philosophy of the reconstruction job matching use-cases in physics analysis. For example, the coordinator of the trigger executable or offline reconstruction does not have to worry about paths interacting with each other. Paths are orthogonal to each other. Since tracking needs to be done before muon finding, putting a tracking module both in the tracking and muon finding paths insures that tracking is available and never done twice. The need to worry about whether tracking has been done or not is never a concern as, in all cases, the complete list of necessary modules is always fully specified.

3) DESIGN SHARING

We have shared with CDF the concepts of paths and sequences. CLEO has provided to BaBar and indirectly to CDF, their concept of Frame and Dynamic Dispatch [Ref. 2 and 3] which we have recoded. Although we have originally attempted to share actual code, we later found that CLEO code had hardened too much to be easily incorporated into the BaBar Framework. Obstacles such as CLEO's reliance on the C++ Standard Template Library (STL) while BaBar staying away from it made the merger difficult. In the end, code designs were shared and through some re-write, we have managed to import significant functionality. We have made sure that the Framework can move either regular physics events or frames whose contents and structure is irrelevant to the Framework. The Framework is said to be "blind" to the frame contents and thus truly earns its designation as an engine. This blindness has made it simpler to share the code between experiments (CDF and BaBar).

4) CODE SHARING AND THE IMPORTANCE OF STARTING WITH A COMMON CODE SET

The Framework code base is shared between BaBar and CDF but not with CLEO. Possibly the most influential factor in the sharing success between CDF and BaBar is that both these experiments use the same file organization and building system called Software Release Tools (SRT) [Ref. 4] and the same code management system, CVS. We have limited the amount of conditional compiles to the absolute minimum. Both BaBar and CDF still maintain separate code repositories. But the format is the same and the CVS code management system has been successfully used to import and merge one version with another. While CVS has definite limitations, some of its key functionality particularly with import and export has been useful. Thus CDF and BaBar can easily share their code base because of the common use of CVS for managing repositories. This is not a statement that CVS is a perfect tool to share code but perhaps more accurately that the tool is the same. We have no direct experience of concurrent development managed either in CVS or other system between different experiments. Some effort in the area of careful exploration into perhaps more modern and possibly more powerful tools is indicated.

It has therefore been possible for CDF and BaBar to inherit new functionality from each other directly at the code level. For example, CDF has contributed to BaBar its parameter menus that allow complex specifications [Ref. 5]. On the other hand, CDF has inherited the Frame concept (as re-engineered into the BaBar Framework from CLEO designs) and, in the same manner, the dynamic dispatch implementation. And CDF has inherited AppAction

objects and its associated Tcl command facility. AppActions allow user-specified special-purpose pre and post event processing to be done.

In the near-term future, we plan to merge CDF and BaBar Framework code again. The immediate gain for BaBar will be all Framework collection classes now relying on an STL implementation. We plan to use an abstract interface to mitigate the effect of adopting STL within the BaBar Framework.

5) MULTI-EXPERIMENT INFRASTRUCTURE COLLABORATION

Code development on the BaBar and CDF code repositories is done in parallel taking extra care that a clash does not result when code mergers are done. This requires frequent electronic communications. But more importantly it requires regular joint workshops. We have had two such joint meetings so far. In addition, we have made use of videoconferencing to maintain the project on track. Of all the types of communications we use, we find that regular workshops have been most fruitful. The common workshops we have held typically had sessions where the emphasis was on defining requirements while the remainder of the time was spend working on actual code.

6) CONCLUSIONS

Our experience in collaborating at the infrastructure-level between evidently different experiments (e.g. a higher cms energy top quark production experiment in a hadron-hadron collision environment compared to a lower cms energy B-meson production experiment in an asymmetric electron-positron annihilation experiment) has been very positive. With CLEO it has been possible to understand and adopt new ways of assembling and distributing aggregates of data from different sources into Frames. With CDF and BaBar, it has been possible to share useful new functionality at an even more fundamental level. We cannot stress enough the importance of early joint design sessions between different experiments. Maintaining constant communication is also paramount.

For all the sharing promises object-oriented software engineering has perhaps made in the last several years through its advocacy of code reuse, our experience has been that more success is attainable with less effort if joint design starts early.

Finally, another point of significant importance is that no single experiment can usually afford a large infrastructure development team. But this team can be assembled by joining forces between experiments with benefits to everyone.

Our experience has been successful at the level of the event reconstruction Framework. We believe the scope of the collaboration can even be expanded to include algorithms although we don't have direct experience to report upon yet. In any case, the benefits of multi-experiment infrastructure collaboration are clear: increased level of productivity in an area that might otherwise be neglected, higher integrity code that results from the thorough checkout necessitated by code mergers and perhaps most importantly an infrastructure forum that benefits from input of requirements of a higher variety than a single experiment could produce.

REFERENCES

- 1) E.D. Frank, Bob Jacobsen, Elizabeth Sexton-Kennedy for the BaBar Computing Group, "Architecture of the BaBar Reconstruction System", presented at CHEP97, Berlin, Germany.
- 2) Paul Avery, Chris Jones, Martin Lohner, Simon Patton, "Design and Implementation of the CLEO III Data Access System", presented at CHEP97, Berlin, Germany.
- 3) Christopher D. Jones, Martin Lohner, Simon Patton, Michael Athanas, Paul , "The CLEO III Data Access Framework", these proceedings.
- 4) Elizabeth Sexton-Kennedy, The FNAL CD/D0/CDF Run II Code Management Working Group. "A Platform Independent Software Management System", presented at CHEP97, Berlin, Germany.
- 5) Elizabeth Sexton-Kennedy and Marjorie Shapiro, "Creating Talk_to Menus for AC++", AC++ Note.