

# The Life Cycle of HEP Offline Software

Peter Elmer, Princeton University

Liz Sexton-Kennedy, FNAL

Chris Jones, Dan Riley,

Valentin Kuznetsov, Cornell University

CHEP07, Victoria, B.C, 6 Sept, 2007

# Offline SW Life Cycle

In addition since the mid-1990s the offline software stack of HEP experiments has been characterized by several themes:

- Predominance of C++ (for new code)

- Geographically distributed collaboration for software development and distributed computing

# Offline SW Life Cycle

Of course the LHC experiments will soon arrive at a critical early point in their life cycle (first data), but they are also characterized by a long gestation period, sometimes with roots stretching back to the same time period in which the running experiments were starting

So (for fun) we thought it would be interesting to go back and compare the evolution of the experiments now nearing the end of their lifetime with one approaching real data taking.

How much code does it take? How much does it evolve after data taking starts? What large changes happened along the way? How many people are involved?

# Methodology

To do this we basically looked back at the CVS history for 4 experiments:

BaBar, CDF Run 2, CLEO III/CLEOc, CMS

We looked in particular at software *releases* over the history of the experiments. This should be a pretty reasonable metric for most of the things the experiments were doing, but it of course underestimates at some level the analysis code.

There was some variation in what is considered a <sup>a</sup>release<sup>o</sup>, including inclusion by value of things that are actually external code. We<sup>o</sup>ve tried to for the most part to eliminate this.

# BaBar

# BaBar

# BaBar

# BaBar



# CDF Run 2

# CDF Run 2

# CDF Run 2

# CDF Run 2

# CLEO III/CLEOc

# CLEO III/CLEOc

# CLEO III/CLEOc

# CLEO III/CLEOc



# CMS

CMS decided in late 2004 to reengineer its application framework, event data model (EDM) and data management, and this effort ramped up during 2005.

Thus its history is divided here into two periods: the <sup>a</sup>classic CMS<sup>o</sup> and the new software (CMSSW)

I will (mostly) focus on the new software

# CMS (CMSSW)

# CMS (CMSSW)

# CMS

# Effort on all of the experiments

# Conclusions (mostly for CMS)

If past experience is any guide:

we may expect that the activity (in terms of number of people working) still has a significant increase to come (50%?)

Half or perhaps more of the code may still be to write (debug, validate, ...)

Many of the lessons learned from the earlier (running) experiments have fed into CMS and, due to the long preparation time, it should avoid some of the problems of very late code delivery (e.g. reco), but where are we fighting the <sup>a</sup>last war<sup>o</sup>? Which new and unforeseen traumas are sitting out there?

We will know more by the CHEP conference after next....

# Afterthoughts...

