SOFTWARE BUILDING

A.N.Dunlop, B.Ferrero Merlino, R.Jones, M.Nowak, Z.Szkutnik

Overview
This track combined software engineering lectures with exposure to the software technologies and packages relevant for LHC experiments. It showed, in a practical sense, how software engineering can help in the development of HEP applications based on the LHC++ software suite and also gave a taste of working on large software projects that are typical of LHC experiments. The lectures provided an overview of LHC++ and covered those aspects of software engineering most relevant for HEP software development.

1. EXERCISES

The hands-on tutorial introduced a series of exercises to solve given problems. The tutorials followed the natural progression of physics analysis exploring the major LHC++ packages on the way. The students completed the tutorials in groups of two.

Essentially, the students were required to develop several C++ programs in succession starting from skeletons:

i) Populate an event database using HepODBMS according to a defined object model.

ii) Build an event tag database from data prepared in I. Select some interesting event attributes and copy them to the event tag database.

iii) Use the Gemini minimisation package to find the minima values for a given set of problems.

iv) Read event tag database built in II and display the contents. Use the LHC++ inter-active graphical tools to apply more cuts.

2. LECTURES

2.1 Track Introduction

Speaker: R.Jones

The goal of this track was to provide an overview of LHC++, which is a comprehensive, mainstream and modern software suite for development of physics analysis software. This overview was coupled with an introduction to those aspects of software engineering that are considered relevant for physics analysis software development. The exercises provided practical hands-on experience of using the major LHC++ packages. The intention was that students should come away with practical knowledge of how to develop physics analysis software in an organised manner. It was not a goal of this track to teach C++, object-oriented concepts, WNT or give details of the physics involved in LHC experiments.

The LHC++ lectures are documented as separate papers in these proceedings. Below is a short summary of the track introduction, closing and software engineering lectures.

2.1.1 Introduction to Software Engineering and OO methodologies

Speaker: A.N.Dunlop

A definition of what is meant by software engineering gave a starting point for this lecture which then went on to explain how the scale of the software project determines the software process required to successfully run the project to completion.

Various processes exist for OO software (OOSE, OMT, Booch, Fusion, Martin-Odell, Unified etc.) and have varying definitions for the phases involved during the project but the Unified process is centred around the architecture and follows a number of iterations driven by use-cases.

2.1.2 An Overview of UML and use-cases

Speaker: A.N.Dunlop

This lecture covered the basic structure of UML, the notation and types of diagrams that can be used to describe the software under development. Emphasis was put on the use cases as a means of driving the development and how they are used at various phases.

2.1.3 Software Design

Speaker: R.Jones

The task of design was introduced as consisting of three levels: architecture, mechanistic and detailed based on the scope of the decisions made. Each level was further defined to show its goals, techniques and deliverables. The UML class, sequence and collaboration diagrams were explained and examples drawn from the exercises. The concept of patterns, how they can be applied to analysis and design and examples from LHC++ packages and the exercises were given.

2.1.4 Software Testing

Speaker: R.Jones

This lecture covered the basic principles of software testing and \why programs have defects. The cost of defect removal and the classification of defects were addressed. The use and basis of software inspections as a means for removing defects was shown to be the most effective way of improving the quality of software. The different types of testing (unit, integration, regression and acceptance) were described and how CASE tools can help in these tasks. The lecture finished with a set of axioms about testing that can improve the way most software developers approach the subject.

2.1.5 Wrap-up on Software Engineering Issues

Speaker: R.Jones

This aim of this lecture was to look at some aspects that affect the long-term well being of development projects. It started by asking three questions:

• Why is the software process so important?

• What is so good about iterative development anyway?

• Why can’t we just get on with writing the code?

To answer the first question, the most common reasons for failure of software projects were listed with how activities, such as adequate analysis and design, can be used to avoid them. The second question was addressed by giving an example of what iterative development means and by showing the unfortunate results of not using it.

Hopefully the students understood that by answering the first two questions the answer to the third becomes clear. As a means of supporting iterative development cycles, configuration management systems were introduced and the lecture finished by emphasising that software always costs something (time or money): either *some* up-front by investing in analysis and design or *more* later to fix all the problems.

2.2 Feedback session

The track finished with a feedback session during which the students asked questions about how to apply the software engineering techniques in different situations, including how to introduce software inspections and how to motivate developers to worry about those issues concerned with software maintenance. There were detailed questions about the use of Objectivity and Gemini in LHC++.

READING LIST

Software Engineering

1. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 4th Edition, 1996, ISBN: 0070521824
2. H.E. Eriksson & M. Penker, UML Toolkit, Wiley, 1998, ISBN: 0471191612
3. M. Fowler, Uml Distilled: Applying the Standard Object Modelling Language, Addison-Wesley, 1997, ISBN: 0201325632
4. Software Engineering Resources http://www.rspa.com/spi/