UCI-Neutrino No. 90-8
SNO-STR 90-79
14 May 1990

Communication Modes During The SNO Software Development And Maintenance

Gerhard Bühler
University of California, Irvine, CA 92717

Abstract

This report is trying to define the communication modes that will occur during the software development and software maintenance for SNO. The definition will have consequences for the way we set up the communication software and infrastructure. The concepts of central software management which are referred to here were outlined in SNO-STR 107-89.

1 Communication scenarios

In the initial phase of the software development different groups are working on separate packages of the software, which are integrated into the central code management system by the librarian as they have been written and initially debugged by the local groups. After this "bootstrapping" phase the different packages together form the pool of software that is centrally managed. From there on, further development adds to the pool as tasks become better defined or more refined. Testing of packages can then be done not only locally but globally within the framework of the complete code. The development phase will see a lot of traffic between the code developers and the central code management system which needs to be standardized and logged.

The code maintenance starts when the code as a whole has been found to test out well and has been accepted as a package. The maintenance phase consists of debugging, adding and modifying existing code. During this phase (which will last throughout the duration of the experiment) all sites need to be automatically updated on new code versions.[1]

This scheme calls for the following functionality of the communication system:

1. Requests to the code management system

   - Listing of files
     A listing of available files or of files which have been updated after a supplied date. Also, lists of files which are currently being worked on can be requested. Files can be grouped and listed in functional units.
   - Extraction of files
     Single files or files from a listing (such as generated by a directory) need to be extracted from the code management system and shipped. For safety reasons, this is for informational or testing use only, i.e. no tag is set in the code management system that the file is being worked on and a modified version of the file cannot be put back into the system.

[1] It may be useful to consider two levels of updates here: preliminary code and finally accepted code.

- History of files
  The history of changes to files with information of who did what when. The information can be summary printout or can be worked into an annotated source listing, which shows the history of lines of code.

2. Requests to the librarian

   - Reservation of files
     A request that a file be marked as being worked on. The file is reserved, extracted and shipped out. During the time the tag is on, no one else can modify the file.
   - Replacement of files
     A request that a file be replaced after having been modified. The file has been debugged locally and can be put back into the code management system for global testing and further distribution.

3. Action by the code management system
   The code management system is triggered to produce a full listing of the code. This can happen in many different ways, such as arrival of new globally accepted code, or passing a calender time mark. The communication system then transports the files to the collaboration sites. There, the code needs to be received and placed into storage which does not interfere with the regular use of the site software (it is conceivable that someone is in the midst of doing development which ties into existing code which then should not be automatically replaced). The communication system places a note into a mailbox informing about the arrival of new code.

2 Requirements for the communication system

The following general requirements for the communication software can be established:

1. Generality
   The system needs to work with different platforms. This makes it necessary to work with a standard protocol such as TCP/IP or BITNET or both for the transfer. The implementation of these protocols has to be well tested.

2. Safety
   No assumptions should be made about the safety and reliability of the networks and the transfer. The system needs to check the successful completion of jobs and to do all the necessary handshaking between the central node and the peripheral sites.

3. Documentation
   The communication system needs to be able to keep a log of traffic that has occurred between the code management system and the users.

3 Suggestion for implementation

¿From tests performed between UCI, Princeton and Penn, and from discussions between Ed Frank, Dick Kouzes and myself, we should investigate establishing a mail server. The server would perform the tasks of a communication system by identifying commands in mail messages and conveying the actions to the respective operating system. The advantages of the system are that it can be inherently very simple and that everyone has mail access by either BITNET, HEPNET or INTERNET. The disadvantage is that the access to the central node would not be interactively, but in batch, which causes some time delay in the distribution of files. However, this time delay can be kept to a minimum by having a monitoring program oberserve incoming traffic. Systems like these have been successfully built before, such as LISTSERV on IBMs, QSPIRES (essentially LISTSERV) at SLAC, and VMSSERV on VAXes at Buffalo.