Planning

Implementation Phases

Our plan of implementation phases. This is in addition to our document of estimated combined effort in ours for each aspect of the project.

Revision History

17/10/01	Initial creation				
		Committed by:	pjm2	Verified by:	tdb1
				Date:	24/03/01
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	

Introduction

In addition to our document on estimated combined efforts in hours for each aspect of our project, here is a more detailed breakdown of each phase of our implementation plan.

Phase 1

Host

The host program will probably be written in basic Java, and simply take the output from running unix commands. This will either be parsed by the program, or more preferably by an external perl/awk/grep/sed program. The thinking behind this is that porting to C/C++ will be much faster if the external parts are already written. This will also allow for the whole basic Phase 1 to be completed easily.

If reading the data is becoming problematic it can be moved to Phase 2 and "simulation data" (i.e. fake) used instead.

It should only be implemented for the main platform, i.e. Solaris/SPARC.

Server

At this Phase the server will only echo output to the screen, and basic checking can be done to verify whether e-mail alerts should be sent (probably via sendmail directly).

Client

No functionality at this stage.

Overall

The aim of this Phase is to implement the basic protocol for communication between the hosts and server, and give us a basis for expansion.

Phase 2

Host

The implementation in Phase 1 should be ported into C/C++ and all data should reflect the actual system (no simulation). It is not necessary to make systems calls at this stage, the external scripts from Phase 1 can still be used.

Server

Database connectivity should be implemented, even if it's just a case of storing rows of data. This will require an element of modularity to be introduced.

Client

A basic web interface should be produced for debugging. All that is required is to be able to view the data in the database to check all is going to plan.

Overall

The aim here is to get the host code into its intended language, and bring the database into the system.

Phase 3

Host

The code should be becoming more modular, with clear sections for the network communications, and the system interfacing. It should now be possible for the program to implement some system monitoring without the aid of external scripts, although if any are proving tricky they can be left for now. The key point is forming a "better" program.

Server

The server should now be taking a very modular design, and the system should be gaining more internal functionality. Better checking of events should be deployed, and the interfaces between components more clearly defined. This will allow for the client interaction in Phase 4 to be more easily implemented.

Client

If time permits, it could be possible to produce some more web pages to allow better viewing of information, although this is the least important part at this stage.

Overall

The system should be taking good shape by now, and still be in an almost working state. Delivery could almost be possible. The main point is that the classes are being clearly defined with a good OO approach in mind.

Phase 4

Host

The host code should be completely standalone at this stage, and capable of running without any outside scripts. It should also be tidied to the point of being finished, and all the proper error handling be implemented. It can be considered complete for the current platform.

Server

The client interface can now be developed to allow a client to connect and view data. This should just be raw data, none of the "thinking" (i.e. filtering of data for clients) need be done.

The collector should also be expanded to allow more "thinking" to occur behind the scenes. Key to this stage is the implementation of a filter, to hopefully stop floods of events going through.

Client

A basic client (in Java?) should be implemented that will connect to the server and just read information. It need not allow much user interaction, and need to act upon events. The key point is the communication between the client and the server.

Overall

The plan here is to polish off the "host side" of the server and get the client side moving along.

Phase 5

Host

With a completed initial host application for Solaris/SPARC (?) porting should take place. Initially NT should be done, but it could be possible for another person to quickly port to the Solaris/Intel and Linux platforms due to the similar code. The NT host application could almost take a rewrite, but all the basic channels of communication should remain the same. C/C++ should be used throughout.

Server

The client interface should be made more advanced with much more "thinking" stuff going on behind the scenes. Preferences may be received from the client to tailor data sent, depending upon the design.

If required, although we expect it won't be, the web interface should be written. However, we expect the webpages will connect directly to the database. Any "live applets" should be able to connect via the client interface.

The server should be effectively complete now, at this stage of the development process.

Client

Client should be fully fledged, allowing full configuration and use a GUI. Graphical representation of the data is desired. This should be considered complete.

The web interfaces should be finished off as well, providing whatever features have been planned.

Overall

Basically bringing everything together. The whole system should be completed at this point. This last phase will probably take quite a while to fully complete (porting and GUI writing).

PUBLIC RELEASE 1

The system is now ready for the initial release. Although testing at each phase should have been done, it is now time to rigorously test the whole system... preferably before release.

Further features could be added, time pending of course, but we have the core of the whole system done.

COMMENTS

This is not final, and should be reviewed with regard to the features list to be produced. It is important that at the end of each phase the following are completed.

- Documentation of code, and thoughts behind implementation.
- Review of code, possibly by others.
- Testing of code.
- Meeting to review progress, and prepare for next phase.

At all times code should be kept in CVS. Tagging will be done at the completion of each phase. It is therefore important that all work be concluded before the next phase commences.

CVS trees should be decided upon at an early stage.