SOFTWARE PRACTICES USED IN THE ESO VERY LARGE TELESCOPE CONTROL SOFTWARE

F. Carbognani, G. Filippi, European Southern Observatory, Munich, Germany

Abstract

In 1999 the first 8-meter telescope of the VLT program is opened to the scientific community. This paper reports on the experience done in applying software engineering practices to the VLT Control Software.

1 INTRODUCTION

The VLT software, now operational at the ESO Paranal observatory in the Atacama Desert, Chile, can be divided into three major areas:

- The VLT Common Software that provides common features used by all developments.
- The Telescope Control Software (TCS).
- The Instrument Control Software, one for each of the 12 instrument, builds on a common frame.

The VLT Software has also been reused for refurbishing other ESO telescopes in the La Silla observatory and it will be used for the remaining VLT instruments, the VLT Survey Telescope (VST) and the VLT-Interferometry (VLTI).

The VLT Software development has involved about 100 software people, roughly 30% ESO and 70% from 30 different institutes and companies, and corresponds to about 200 man-years, 50% done by ESO and 50% by externals.

It is possible now to report about the application of Software Engineering practices as they have been announced at the beginning of the project [1].

2 DEVELOPMENT ENVIRONMENT

The key rules have been:

- Dividing the whole development into software modules. Each module is a set of files organised in a fixed directory structure and has a unique name. Figure 1 gives the modules divided by VLT Common software and applications.
- Structuring make usage: a Makefile for each module defining what has to be processed, and a centralised vltMakefile taking care of what specific to operating systems and languages and providing a set of standard actions (generation of automatic dependencies, clean, man pages, install).
- Homogenising programming style by applying Coding Standards, Templates, and Naming Conventions. This makes support easier and allow better code sharing between people.

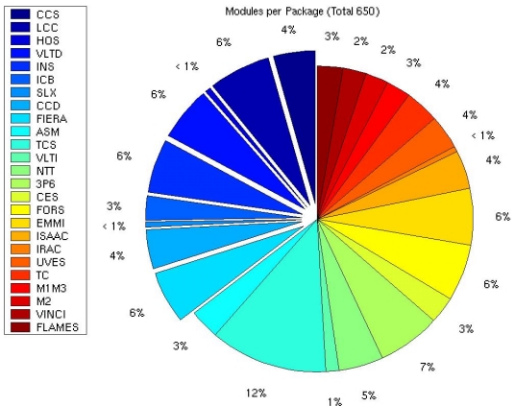

Figure 1 - Modules on the VLT Software Archive

- Selecting and supporting the development tools (GNU, tcltk, etc.)
- Standardising the UNIX environment set up of every VLT user, both development and operational ones, and keeping trace of the configuration files needed to customise the operating system (OS) and tools.
- Defining a development environment based on user areas, integration areas and released code area. The current active areas are pointed by environment variables, the usage of make and the UNIX environment set up are done accordingly.
- Packing and distributing the development tools and the VLT Common Software as a "commercial" product, i.e., planned releases, twice a year, installation instructions covering from the OS to the VLT code, manuals. In Figure 2, the Lines of Code (LOC) of the latest releases.

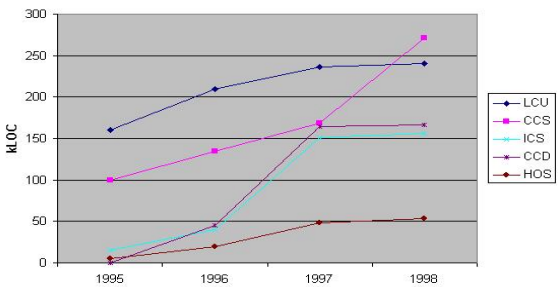

Figure 2 - LOC for VLT Common Sw

Making better: maintaining the rules, providing more tools for automatic check, using systematic code review (done in the past, but on a sporadic way).

3 CONFIGURATION MANAGEMENT

The two pillars of the VLT Software Configuration Management are the code archive and the VLT Software Problem Report (VLTSPR) procedure.

The code archive supports code configuration during development AND integration on geographically distributed sites. The design keys have been:

- The configuration item is the software module. This allows a reasonable but still simple flexibility in system configuration (a system is made up of 15 to 100 items, identified by their name and version, corresponding to 2000 to 20000 files).
- There is only one central archive. Users get local copies using a simple client server mechanism. In projects as we do, typically there is only one person at time dealing with a specific part. A modification must be started, implemented and tested, then archived. During this period the module is LOCKED.
- To allow experiments or patches on older versions, branches are also supported.

The code archive is implemented using RCS and a set of ad hoc programs and scripts (cmm) implementing the client-server interaction and the user interface. The archive is physically located at the headquarters and used also by teams in Chile and several institutes.

Figure 3 gives the accesses per month for modification and for read only mode. Currently approximately 60% of

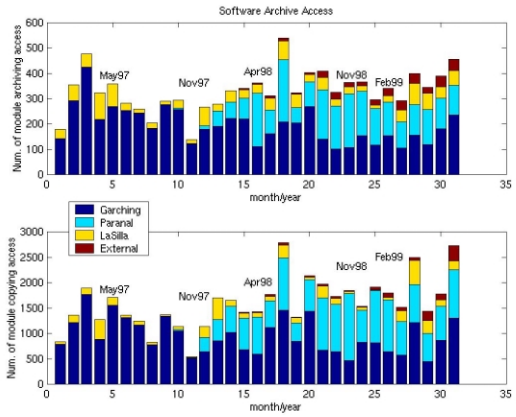

Figure 3 - Accesses to VLT Sw Archive

the accesses are from outside the headquarters and 25% from non-ESO sites.

The central archive is not only the software repository, but it is also a management tool. In archiving a module, the developer tells all the other people "Hey, there is a new CONSISTENT and TESTED set of files that you can use!". This is very useful when several teams operating on different time zones do development and integration. Comparing the current configuration against the status of the archive, the integration responsible can see that new items have been produced. A glance to the comments stored by the developer to qualify the newly archived version is normally enough to decide whether to take the new version or not. In this way NTT, VLT, instruments have been developed, integrated and commissioned on top of mountains at 2500 m in the middle of a deserts, but involving people in several countries on both sides of the ocean.

The VLTSPR System is meant to be used by both internal and external users of VLT software to report errors in code or documentation or to propose a change. VLTSPR is build using the commercial tool Action Remedy (c) and has a Web Browser interface. This is the basic workflow of the system:

- Problem submitted, depending on the subject, some people are notified immediately
- The SPR is discussed in the Software Configuration Control Board meeting and a Responsible Person is appointed for the problem
- Responsible works on problem
- Responsible can close the SPR (must add a final remark on it)
- People can add comments any time

At present there are about 120 names in the user database, 75% ESO users (both Europe and Chile), the remaining 25% from 13 external projects, some project have more sites.

In figure 4 the trend of the SPR archive over the years. The number of open SPR has been always kept under a physiological limit (about 500) that corresponds to what we are able to treat between two releases.

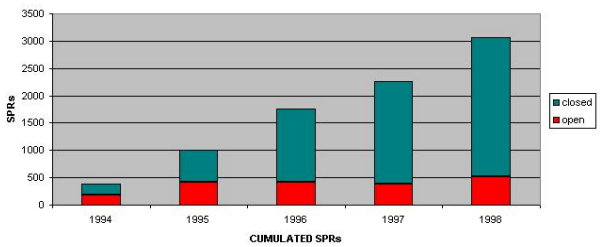

Figure 4 - Cumulated SPRs

Figure 5 reports the distribution of the SPR per area in '97 and in '98.

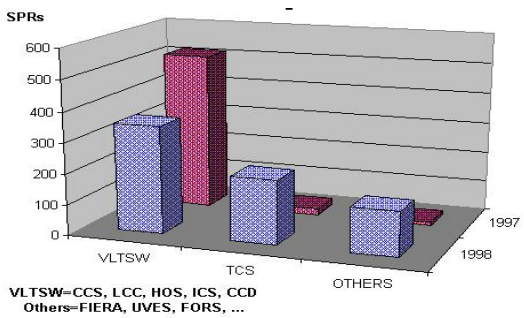

Figure 5 - SPRs per Different Areas

As the project evolved, SPR concerning the common software are decreasing, while the one for the application

part are going up, sign of the integration activity taking place.

Making better: we are developing a better client-server mechanism and we plan to make the cmm software available on the public domain. No further developments planned on VLTSPR.

4 DOCUMENTATION

The documents produced according to the VLT Software Management Plan are of the following types: PLA for plans (management, configuration management, tests), SPE for specifications (requirements, functional, design description), VER for test documentation, MAN for manuals, TRE for technical reports, SOW for statement of work, PRO for procedure (standards), INS for instruction (standards)

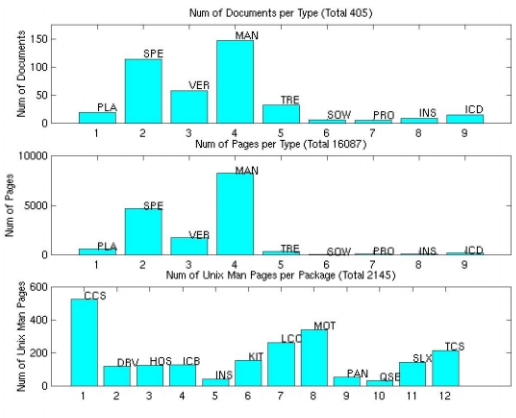

Figure 6 - VLT Sw Documents Statistics

Documents have been developed mainly using FrameMaker and the files are stored on central repository. Configuration Control is done manually. Paper copies are managed by the central VLT Project Archive. In addition there is an on-line documentation in form of man-pages. Every function, command, utility has its own man-page that is part of the source file and is automatically extracted by the make man command.

Practically the totality of the specifications have gone through at least one formal review (draft distributed to a panel of reviewers, written comments from reviewers, review meeting for discussing the comments, document approval or new loop).

Figure 6 shows the total amount and the A4 pages per document type and the number of manpages.

It is possible to notice that we concentrated our effort on the specification and on the final documentation, tests documentation is reduced because a relevant part of the software is covered by automatic testing. The number of man-pages can be read as the "functional points".

The VLT Software documentation is available from http://www.eso.org/vlt/vltsw

Making better: due to time and resource constrains, we did not use the WEB technology for on-line documentation. It has to be done in the next future.

5 AUTOMATING TESTING

As "what cannot be observed cannot be controlled", what cannot be tested cannot be successfully delivered. An the only way to deal with testing is to have automatic testing that implies:

- The test execution has a standard interface so everybody can run everybody else tests.
- The test command prepares the environment set-up, executes the test(s), filters out variant data (dates, time, host names), compares results against reference and provides FAILED or PASSED.

To support automatic test on the VLT Software a Tool for Automated Testing (tat) has been developed.

Figure 7 show the present Automatic Test coverage on the VLT Common Software.

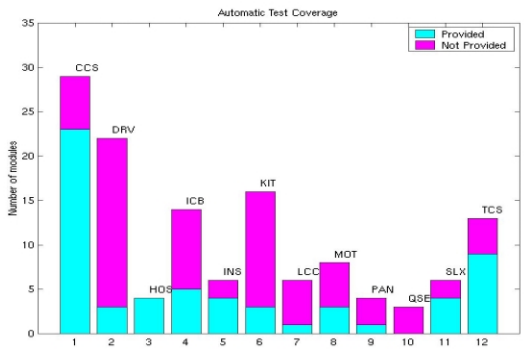

Figure 7 - Automatic Test Coverage

Making better: increase the number of modules using automatic tests, improve tat tool to handle more situations and better stubs.

6 FUTURE DEVELOPMENT

Beside maintaining and improving the existing procedures, our next goals are:

- Use RUP/UML as analysis and design methodology. A pilot project is on going [2]
- Support Java
- Use WEB as on-line documentation tool
- Use more PC-based tools
- Base test procedures on "use cases"

7 REFERENCES

[1] G.Filippi, “Software Engineering for ESO's VLT project”, ICALEPCS '93.

[2] G.Chiozzi J.M.Filgueira, "Real-time Conrol Systems: a One Document OO Development Process", ICALEPCS99