SOFTWARE ENGINEERING FOR THE VIRGO PROJECT AT EGO

F.Acernese[6], P.Amico[10], M. Alshourbagy[11], S.Aoudia[7], S. Avino[6], D.Babusci[4], G.Ballardin[2], F.Barone[6], L.Barsotti[11], M.Barsuglia[8], F.Beauville[1], M.A.Bizouard[8], C.Boccara[9], F.Bondu[7], L.Bosi[10], C.Bradaschia[11], S. Birindelli[11],S.Braccini[11], A.Brillet[7], V.Brisson[8], L.Brocco[12], D.Buskulic[1], E.Calloni[6], E.Campagna[3], F.Carbognani[2],F.Cavalier[8], R.Cavalieri[2], G.Cella[11], E.Chassande-Mottin[7], C. Corda[11], A.-C.Clapson[8], F.Cleva[7], J.-P.Coulon[7], E.Cuoco[2], V.Dattilo[2], M.Davier[8], R.De Rosa[6], L.Di Fiore[6], A.Di Virgilio[11], B.Dujardin[7], A.Eleuteri[6], D.Enard[2], I.Ferrante[11], F.Fidecaro[11], I.Fiori[11], R.Flaminio[1,2], J.-D.Fournier[7], O.Francois[2], S.Frasca[12], F.Frasconi[2;11], A. Freise[2], L.Gammaitoni[10], A.Gennai[11], A.Giazotto[11], G.Giordano[4], L. Giordano[6], R. Gouaty[1], D. Grosjean[1], G.Guidi[3], S.Hebri[2], H.Heitmann[7], P.Hello[8], L.Holloway[2], S. Karkar[1], S.Kreckelbergh[8], P.La Penna[2], N.Letendre[1], V.Loriette[9], M.Loupias[2], G.Losurdo[3], J.-M.Mackowski[5], E.Majorana[12] , C.N.Man[7], M. Mantovani[11], F. Marchesoni[10], F.Marion[1], J. Marque[2], F.Martelli[3], A.Masserot[1], M.Mazzoni[3], L.Milano[6], C. Moins[2], J.Moreau[9], N.Morgado[5], B.Mours[1], A. Pai[12], C.Palomba[12], F.Paoletti[2;11], S. Pardi[6], A. Pasqualetti[2], R.Passaquieti[11], D.Passuello[11], B.Perniola[3], F. Piergiovanni[3], L.Pinard[5], R.Poggiani[11], M.Punturo[10], P.Puppo[12], K.Qipiani[6], P.Rapagnani[12], V.Reita[9], A.Remillieux[5], F.Ricci[12], I.Ricciardi[6], P. Ruggi[2], G.Russo[6], S.Solimeno[6], A. Spallicci[7], R.Stanga[3], R. Taddei[2], M. Tonelli[11], A. Toncelli[11], E.Tournefier[1], F.Travasso[10], G. Vajente[11], D.Verkindt[1], F.Vetrano[3], A.Viceré[3], J.-Y.Vinet[7], H.Vocca[10], M.Yvert[1], Z. Zhang[2], J. D. Wet[2]

[1]Laboratoire d'Annecy-le-Vieux de physique des particules, Chemin de Bellevue - BP 110, 74941 Annecy-le-Vieux Cedex – France

[2]European Gravitational Observatory (EGO), Via E. Amaldi, I-56021 Cascina (PI) Italia

[3]INFN – Sezione di Firenze/Urbino, Via G.Sansone 1, I-50019 Sesto Fiorentino and/or Università di Firenze, Largo E.Fermi 2, I - 50125 Firenze, and/or Università di Urbino, Via S.Chiara, 27, I-61029 Urbino, Italia

[4] INFN, Laboratori Nazionali di Frascati, Via E. Fermi, 40, I-00044 Frascati (Roma) - Italia

[5]LMA, 22, Boulevard Niels Bohr, 69622 - Villeurbanne- Lyon Cedex, France

[6]IINFN – sezione di Napoli and/or Università di Napoli "Federico II", Complesso Universitario di Monte S. Angelo, Via Cinthia, I-80126 Napoli, Italia and/or Università di Salerno Via Ponte Don Melillo, I-84084 Fisciano (Salerno), Italia.

[7] Departement Artemis - Observatoire Cote d'Azur, BP 42209, 06304 Nice , Cedex 4, France

[8]Laboratoire de l'Accélérateur Linéaire (LAL),IN2P3/CNRS-Université de Paris-Sud, B.P. 34, 91898 Orsay Cedex – France

[9] ESPCI - 10, rue Vauquelin, 75005 Paris - France

[10]INFN Sezione di Perugia and/or Università di Perugia, Via A. Pascoli, I-06123 Perugia – Italia

[11]INFN - Sezione di Pisa and/or Università di Pisa, Via Filippo Buonarroti, 2 I-56127 PISA – Italia

[12] INFN, Sezione di Roma and/or Università "La Sapienza", P.le A. Moro 2, I-00185, Roma

Corresponding author e-mail address: franco.carbognani@ego-gw.it

ABSTRACT

The VIRGO Gravitational Waves Detector is continuing its commissioning phase. An important element in this phase is the application of Software Engineering (SE) practices to the Control and Data Analysis Software. This article focuses on the most recent achievements in applying and evolving those SE practices as a simple but effective set of standards and tools. The main areas covered are software configuration management, problem reporting, integration planning, software testing, documentation and systems performance monitoring. For each of these key areas a specific tool has been developed or customized, respectively: SCVS (developed on top of CVS) for configuration management, Software Problem Reports (SPR, implemented on top of the WREQ tool) for tracking bugs and change requests, automated test tool (TAT) for software testing, Doxygen for documentation and BigBrother for system performance monitoring. Efforts are underway in building on top of those simple tools an Integrated Development Environment (IDE) and in introducing tools and procedures that will aid in the analysis and design phase. This includes templates for writing requirements and

design specifications as well as making use of UML as standard modelling language. The final goal is to effectively support every phase of the Software Development Life-Cycle.

THE VIRGO PROJECT

The VIRGO project consist of a suspended Michelson Interferometer with two 3 Km arms aimed at the detection of gravitational wave signals from cosmic sources [1]. The VIRGO Control Software provides the VIRGO installation of all control and monitor functions and can be summarized by the following values:

700933 Lines of Code (LOC) of C/C++ and Assembler (See Figure 1)

50 Workstation + 60 Local Control Units (RIOs) for the Interferometer operation

2 countries, 6 laboratories, 18 teams, almost 100 developers over the current lifespan of the project.

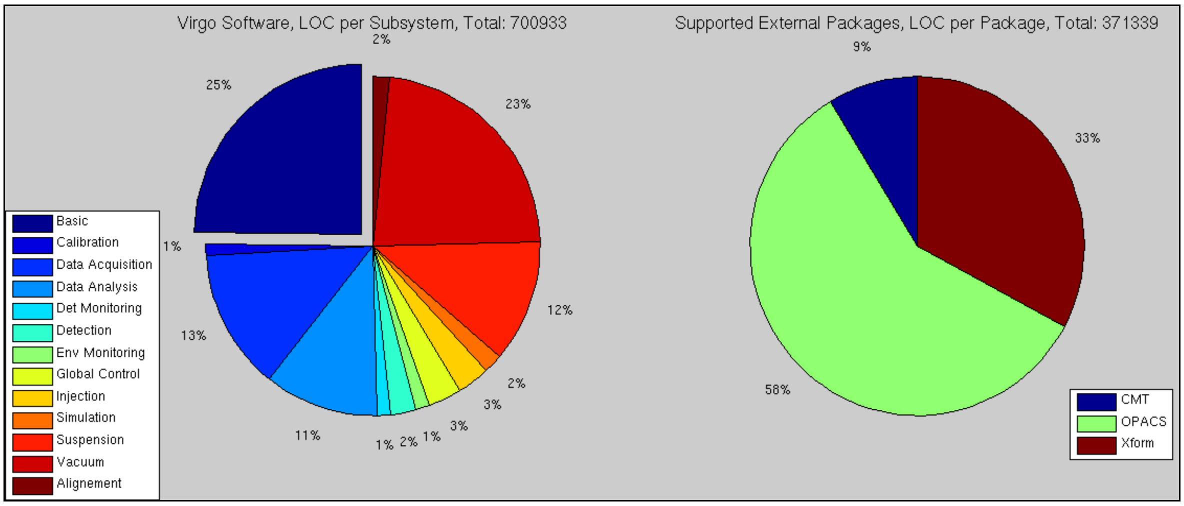


Figure 1. Lines of Code (LOC) of the Virgo Software and the Supported External Packages

SOFTWARE CONFIGURATION MANAGEMENT

A tool called SCVS has been developed on top of CVS in order to provide an easy and structured approach to software configuration management. It manages packages more than files, and blocks the access whenever a package is being modified. By archiving a package the developer tells the others that a new consistent set of files is ready for use.

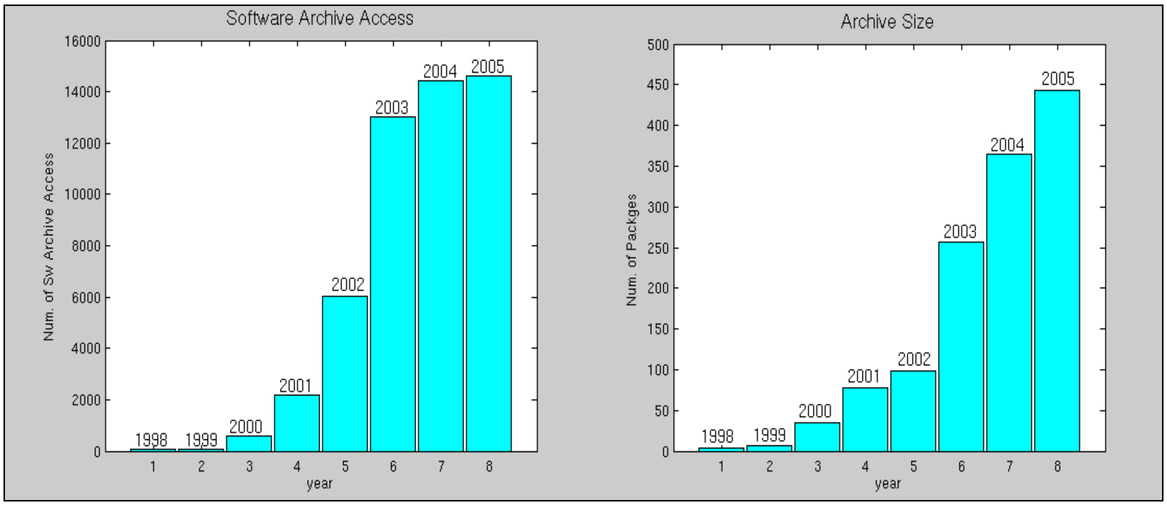


Figure 2. Virgo Software Archive Access and Size

Just a glance at the comments is usually enough for the others to decide whether to take the new version or not. SCVS does not currently prevent archive access via native CVS command.

Experienced CVS users can work directly at CVS level and even in this case the workflow underlined by SCVS can still be followed. Figure 2 shows the total number of CVS/SCVS transactions executed per year and the growing in size of the software archive in terms of number of packages over the last eight years. For 2005 the data are up to the month of August. It is possible to see how the introduction of SCVS in February 2003 coincides with a steep increase of the archive use and size.

SOFTWARE PROBLEM REPORT (SPR)

For the VIRGO SPR system the tool WREQ has been chosen and customized in order to handle software change requests and problem reports. Figure 3 shows the trend of the SPR archive as of middle of September 2005. It is possible to see that the tool is finally leaving its start-up phase and getting wide adoption since patterns like the release cycle can be identified in the graph. Statistics like these could provide indicators useful for the evaluation of the software maintenance effort.

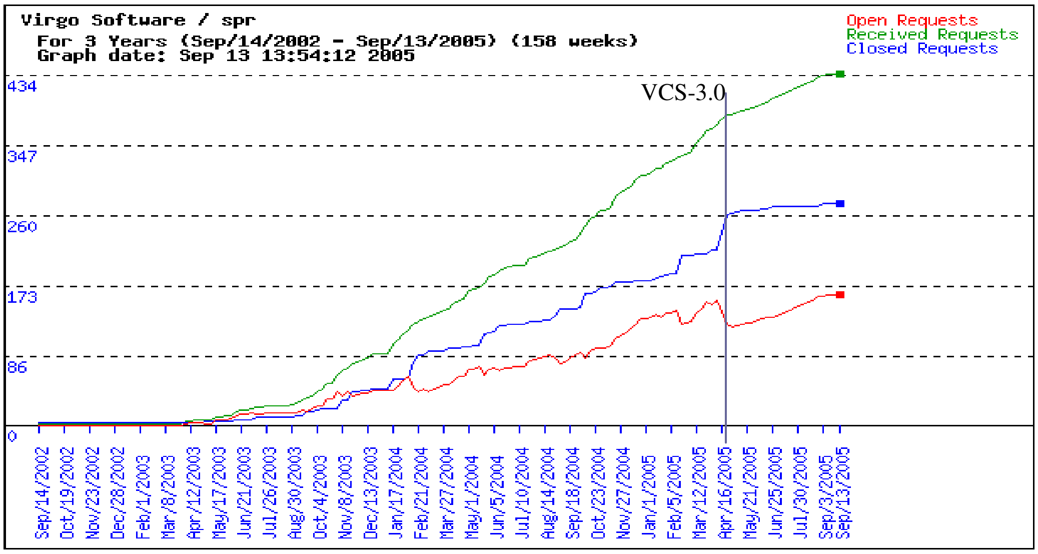


Figure 3. SPR Statistics

DEVELOPMENT ENVIRONMENT

In order to deal with a huge amount of developers spread over several geographic locations the tool CMT [6][7][8] is used to provide a standard common environment. It provides features like, e.g., a structured make usage, easy encapsulation of third party tools and libraries, and standardization of the users UNIX environment set-up. The development environment is completed by the definition of a common development area, an integration area and a released code area.

DOCUMENTATION

For documentation the main effort in this area focused mainly in supporting documentation that can be extracted from the code itself. The tool selected for this purpose is Doxygen [9], which extracts the information from the commented code and creates reference manuals with different levels of detail depending on the tags used in the commented code and on the configuration file Doxyfile. Specific CMT fragments have been made available for the generation of the Doxygen documents via "make" and a Doxyfile template has been provided.

TESTING

To support automatic tests on the VIRGO software the Tool for Automated Testing (TAT, originally developed at ESO for the VLT project [2]) has been adopted and slightly modified for the VIRGO software specificities. This tool is easy to use and configure, it is able to prepare the environment, execute the tests, filter out variant data (dates, time, host names), compare results against reference and provide a final FAILED or PASSED as result. On the other hand, to provide a system simulator the VIRGO Test Facility (VTF) has been set up. It has been designed to permit the simulation of the hardware and software environment and can so be used for integration tests, measurement of performance, etc. It is used to validate new releases before they are put online and also to reproduce problems, being so an important element for the Interferometer commissioning and science phases.

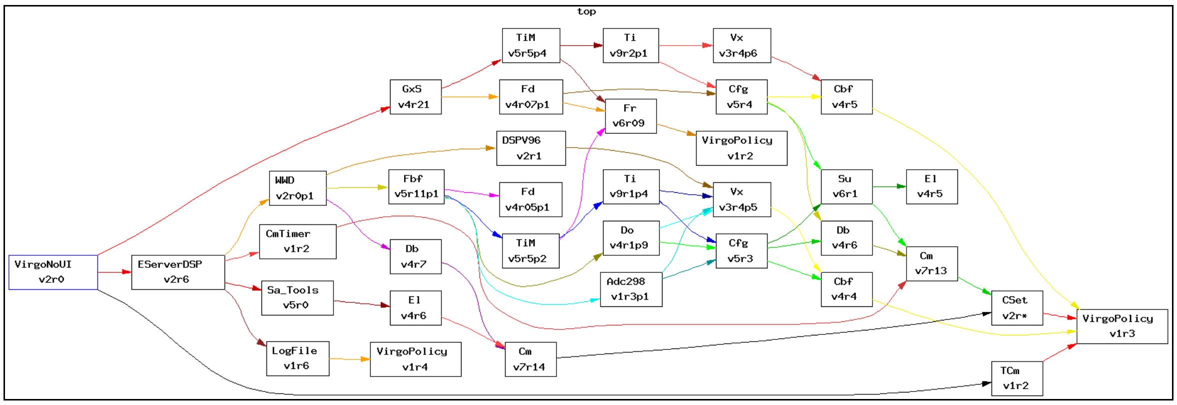


Figure 4. CMT Dependency Graph for VCS-1.0 (Package VirgoNoUI)

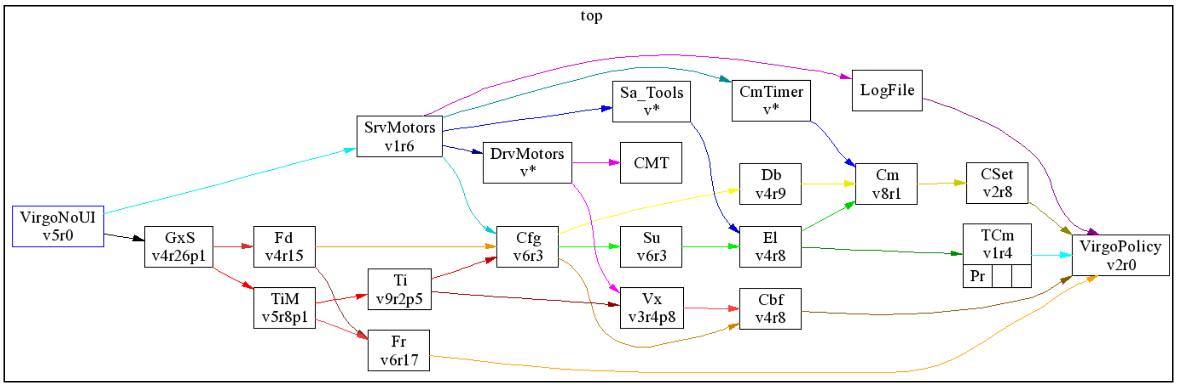


Figure 5. CMT Dependency Graph for VCS-3.0 (Package VirgoNoUI)

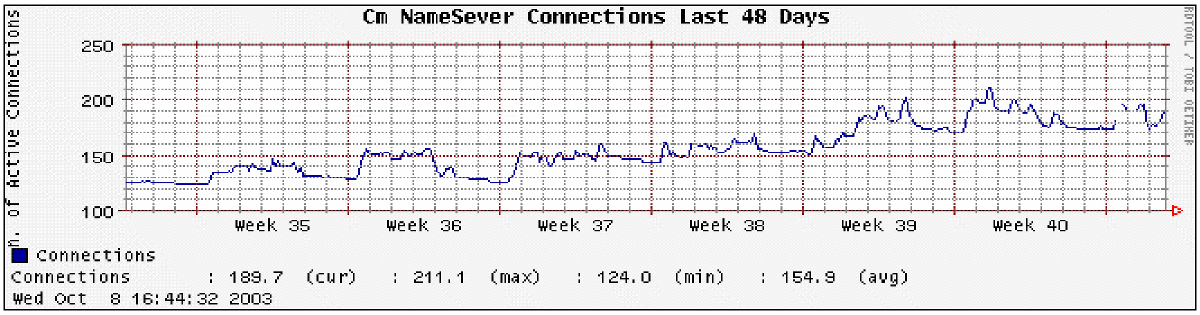
VIRGO SOFTWARE RELEASES

One of the major problems on software development is the software integration. At VIRGO the integration plan is based on a slow upgrade cycle of 3-6 months for major (external) releases, and on demand for minor (internal) releases. Each major release has a specific configuration in terms of platform, e.g., the operating system is part of the release baseline. For each software release the supported configuration in terms of hardware and software are detailed at a dedicated Web page. The VIRGO Common Software includes also third party software used in the development process like, e.g., GNU tools, Java, Tcl/tk and support packages like root or FFTW. From the release baseline three software distributions are generated: two containing data analysis/simulation tools, and the third supporting data analysis users at the VIRGO Computing Centers (CNAF in Bologna and CCIN2P3 in Lyon). The introduction of Software Releases is already giving tangible results. Before its introduction the software was more or less periodically installed in the integration area without a complete realignment process. This mechanism was very flexible since any software could be installed at any time, but at the price that old libraries required by not fully-aligned packages were still in use. The first VIRGO Software Release VCS-1.0 was still suffering this problem and could be put together in reasonable time only by compromising on such aspect. All of this is clearly shown on Figure 4 by the CMT use dependency graph among some of the packages running on the Local Control Units.

Figure 5 show the same relationships for the latest VCS-3.0 Release. It could be seen that no more than a single version of each package is present. This is true for the whole VCS-3.0 release software meaning it implement a fully aligned baseline

SYSTEM MONITORING

The monitoring of the different VIRGO subsystems has been identified as crucial for the commissioning phase. For this purpose the tool BigBrother (BB) is used. BB is a widely used tool for the monitoring of system and network services. For VIRGO the scope of BB has been extended to the tracking of failures and alarm management. For example, custom scripts have been developed and integrated in BB that permit to monitor the Cm name-server (being Cm the layer providing interprocess communication), the database server and the error logging system. With those BB extensions it is then possible to provide alarms on essential VIRGO common services, monitoring their performance and robustness.



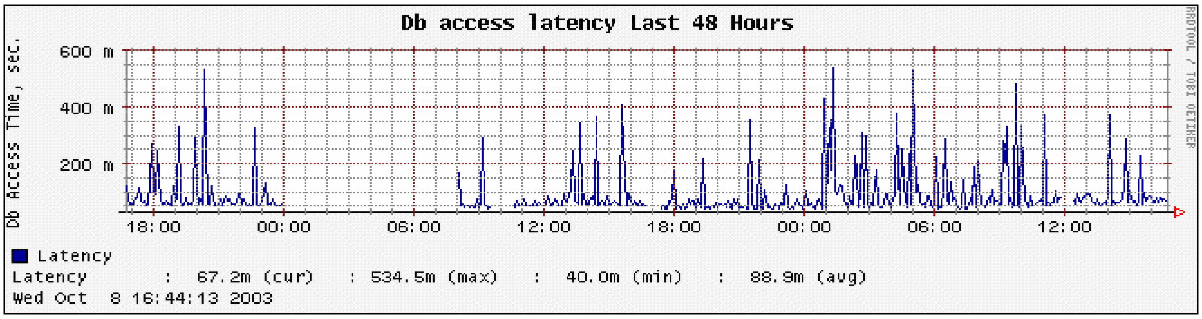


Figure 6. Big Brother trend graph for the Virgo Cm Name Server and Database Server

As an example Figure 6 shows the number of active Cm-connections and database access latency. Those graphs have been very effectives in spotting specific software problems.

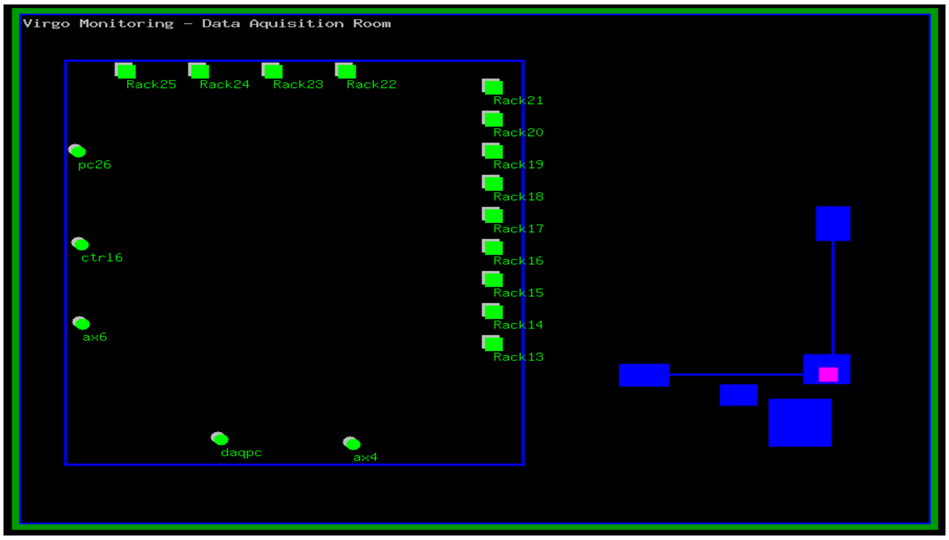


Figure 7. DAQ Room BBMAP display

The tool is currently being enhanced with new tests like, e.g., the monitoring of the most critical servers of the data acquisition chain. A new feature has also been added via the BBMAP plug-in. BBMAP is a set of PHP pages that give a status map capacity to Big Brother. With this functionality is possible to link the error information with the location of the involved hardware. This is facilitating the problem diagnosis and speeding up intervention time. Figure 7 show a BBMAP screenshot detailing the position of workstations and racks on the Data Acquisition Room.

THE FUTURE

All described SE practices are currently in place and getting part of the daily routine. To further improve and evolve these practices several pilot projects are currently carried out like, e.g., the Noise Analysis Package (NAP) [10]. Within the context of the NAP development an effort is made to evolve to an object-oriented approach. This includes the definition of templates for requirements analysis and design documents based on UML models, C++ class templates with Doxygen tags, and the usage of tools like Umbrello [11]. Also the incorporation of the Web as platform for documentation, knowledge base and shared workspace is a major goal. For this purpose the Wiki tool has been set up and is currently being evaluated. Another topic of interest is an Integrated Development Environment (IDE). For the moment the tool KDevelop [12] has been customized for fully automated Doxygen documentation generation and for CMT integration. Higher level of integration with the existing tools is being studied. Also different IDEs are under evaluation. The NAP project has already reached a good level of maturity at the mentioned areas and the feedback from users is generally positive.

CONCLUSION

The VIRGO SE Practices represent a balance between heavyweight existing standards and pragmatic choices. The approach taken for implementing these practices was to embed them in a set of consistent tools rather than enforcing a set of rules to which developers must adhere to.

Scientific software projects like VIRGO are becoming too large and complex to be managed without proper SE procedures. Obviously SE does not come for free, but our experience demonstrates that it can be brought into the project at a reasonable cost. We have chosen to implement SE in gradual steps, providing first the most urgent tools and procedures, and then let focused pilot projects lead the way for the “full SE approach” that we will improve and evolve continuously.

REFERENCES

[1] S.Braccini et al., “Status of Virgo”, Proc. of Amaldi 6 Conference, August 2005.

[2] G.Filippi, “Software Engineering for ESO's VLT Project”, ICALEPCS 1993.

[3] G.Filippi, F.Carbognani, " Software practices used in the ESO Very Large Telescope Control Software", ICALEPCS 1999.

[4] G.Filippi, F. Carbognani, P. Siviera, “Software Engineering Practices for the ESO Very Large Telescope”, ADASS Conference, Boston USA, November 2000.

[5] F.Carbognani, Jacques de Wet, “Software Engineering Practices for the EGO Virgo Project”, SPIE Astronomical Telescopes and Instrumentation 2004 Conference, Glasgow Scotland, June 2004.

[6] http://www.cmtsite.org

[7] C. Arnault, "CMT, a software configuration tool", CHEP2000 (paper F033).

[8] C. Arnault, B. Mansoux, A. Pérus, "Experiencing CMT in Software Production of Large and Complex Projects. Issues in the scalability of software production management", CHEP01

[9] D. van Hee, “Doxygen”, http://www.doxygen.org

[10] E. Cuoco et al, “NAP: A Tool for Noise Data Analysis. Application to Virgo Engineering Runs”, Procedures of GWDAW2004, March 2005.

[11] P. Hensegen, “UML umbrello modeler”, http://uml.sf.net

[12] http://www.kdevelop.org/