

AdaCore Quality Procedures

Romain Berrendonner
Software QA Manager, AdaCore
berrendo@adacore.com

Introduction

GNAT Pro is a tool-chain for the Ada 95 language including a compiler, an IDE, and a number of specific tools, like a code pretty printer, a memory check program, a stack usage analysis tool and many others. GNAT Pro has been a free software from its origin as a New-York University project. It is now mostly supported by AdaCore, a company present in both the United States and Europe, which provides services for more than 450 customers all around the world in the aerospace, defence, air traffic control, health-care, banking and others industries industries.

The compiler included in GNAT Pro is based on the GCC technology. Like other compilers part of the GNU Compilers Collection (GCC), the front-end is language-specific, while the back-end is in charge of generating executable code in a language-independent fashion.

If GNAT Pro leverages on the general community effort around the GCC backend, it is also one of the prominent contributors to this effort with several people working full time on improving the backend. The GNAT Pro Ada front-end relies on a team of developers from a single company, distributed in different remote places, including France, United States (East and West coast), Russia and Spain, while the number of contributors external to AdaCore remains small. Even if the process for submitting patches is as simple as sending e-mail, in practice only a limited number of individuals contributed changes to the GNAT front-end or runtime, and their changes were easy to handle. Like the front-end, the other tools part of GNAT Pro are mostly supported by AdaCore, eventhough its large customer base provides a significant and continuous feedback.

This situation is fairly original in the free software world. Most free software are developed by informal communities of individuals or companies, often linked very loosely together. However, the issues raised by the geographical distribution of people within AdaCore can be similar to those such communities are facing, and most of the techniques used by AdaCore to handle regular software development and release cycles could be used by other free software communities. This paper describes these methods and procedures, which have been in force at AdaCore for several years and are being continuously enhanced.

In particular, producing high-quality releases is a technical challenge. Tens, sometimes hundreds of specific issues have to be identified, tracked and understood before any conscious decision can be made about them. For this to happen, one need to have a controlled development process, which allows to track such issues from black board up to coding and validation. Customer concerns have to be taken into account too when issues are prioritized. Last but not least, timing and resources are important issues too. This paper will first present the human challenges in this process and how they can be addressed (1) and then go through the technical issues (2) related to this process, before examining in greater details the future of AdaCore's infrastructure and procedures (3)

1. Human challenges

Vérité en deçà des Pyrénées, erreur au-delà (What is true in France is wrong beyond the Pyrénées) said the great French writer and moralist Blaise Pascal. The same holds for human interaction in

multi-national companies. Language and cultural habits can cause communicational difficulties not always easy to solve. E-mail based discussion, with its specific pattern of immediate written communication, can very easily lead to misunderstanding and therefore to technical mistakes that turn into software bugs.

The huge amount of information can also be an issue. Every Internet user receives dozens of spam messages a day. Corporate users get in addition their daily dose of marketing and technical messages, in which, from time to time, a very important piece of information is hidden.

AdaCore employees have an even more demanding environment. Every change in the sources is sent to interested developers; specific technical discussions take place on specific mailing lists; the automatic build and testing infrastructure described later in this paper sends periodic notifications to signal malfunction or send out testing results. Overall they receive several hundred messages a day.

In order not to avoid information to be lost in this maelstrom of information, AdaCore has developed a tracked mailing list system which gives persistence to the volatile e-mail data. Every technical discussion, no matter whether it is a customer support request or an internal design discussion, is given a unique reference number, known internally as a ticket number. When a message holding such a ticket number on the subject line is sent to a mailing list, it gets automatically stored in an internal database which can later be used as a knowledge base. A web interface allows internal users to browse these ticket numbers, but also to prioritize them, assign them to a particular employee for a specific action, or mark them as to be included in the next release.

Even if it is good to make sure that no information is lost, it is even better to make sure that people do not forget to carry out their specific task. To do this, a very important issue in a company which does most of its business with support services, a reminder service is hooked with the ticket number database, which sends periodic messages to employees with their list of open issues. Any non-assigned ticket number is sent to the management, which is in charge of assigning them.

All of this makes communication within AdaCore very efficient, despite the human and cultural challenges it faces every day. Technical issues remain however the most challenging ones.

2. Technical challenges

AdaCore ships two main releases a year, on more than 40 different platforms. This process is a demanding one as the time constraints, which is led by internal resource consumption and strong customer demand, are tight. It is also a delicate process, with release stability a paramount for a company providing development solutions to customers in sensitive areas. The general release process is discussed first (A) and the development process second (B).

A) The release process

In order to meet those challenges, AdaCore has developed an original release process, which starts way before the actual software is built. This process relies on a branched set of sources, so that development is a little impacted as possible by release activities, and conversely.

In a first phase, the ticket numbers of interest go through a specific cycle which aims at identifying whether they should go into the release. They are first marked as «PENDING », until a decision is made by the release committee on technical grounds after extensive discussion. When a decision is made, the ticket numbers are marked either as « IN » if they must be included or « OUT » if not. Later, ticket numbers marked for inclusion are checked into the release branch and therefore marked

as « MERGED ». During all this process, a reminder mechanism is used to ensure that no ticket number is overlooked. All members of the release committee get a daily list of ticket numbers in the « PENDING » state, so that they know what is left to be discussed. Developers get a list of the ticket numbers assigned to them in « IN » state that they must merge and mark as « MERGED » when done.

When all ticket numbers have been considered and when all those marked for inclusion are actually merged into the release branch, the release build can start. The build includes not only the construction of software but also the execution of all the test-suite included in the AdaCore quality plan:

- the internal regression suite built out of customer test cases, which now comprises more than 12.000 test cases;
- the ACATS test-suite, which provides conformance tests to the Ada standard;
- a set of specific test-suites for particular add-ons or tools, like the debugger, the IDE, the distributed programming solution and so on.

Those reports all get carefully analyzed by dedicated specialists. Any unexpected failure at this stage must be properly understood, and should it be a critical one, a new ticket number is opened with appropriate priority so that a new build can be done with a fixed set of sources. This situation is a rare one, though, because of the preventive measures described below.

A special word on the GCC back-end selection process. It is a delicate issue to select the very version of the back-end to be used for a particular GNAT Pro release. AdaCore now tries to use the most recent GCC technology to take advantage of the advances in compilation technology. However, it has price to pay in terms of stability and, as we said, stability is a paramount for GNAT Pro users. To find a balance between those opposite issues, AdaCore does not use the head version of GCC but the most recent stable version. The changes on this branch are monitored closely, and when necessary, special patches are applied to this branch so that regression found by the internal testing process are fixed. Later on, these special patches are submitted by AdaCore to the FSF, so that GCC takes advantage of our findings as well.

After all automated checks are validated, a last testing phase is carried out to make sure that the whole process went smoothly. It involves a manual check – the sanity checking in AdaCore's slang – where assignees go through a test book to make sure that all automation did not slip a major defect. This testing is limited, because it is manpower-intensive. Therefore only things that are not tested automatically are part of the sanity checking.

The releases of GNAT Pro are therefore carried out in a very centralized fashion, with a release committee taking the most important decisions, and a little team of release engineers enforcing them. However, the release process cannot be disconnected from the development process, and the latter has to be as sound as the former.

B) The development process

Actually, the release cycle is the end of the development work and would be soon completely unmanageable if proper development procedures were not put in place and harshly enforced.

Such procedures start with the design stage. When a new feature is required or a significant bug fix, a new ticket number is opened as a reference. The design discussion is stored there; they take place on a mailing list where all engineers belong, so that the widest possible input is gathered. As soon as consensus is obtained, implementation can start.

The implementation process includes several checks that ensure not only the correctness of the fix or feature implementation but also that it is correctly inserted in a wider context:

- Before being committed in the configuration management system, all changes must be tested. A specific infrastructure has been specifically developed for this. Internally known as « the mailserver » this facility receives specifically formatted e-mails with the patch to be tested, applies the patch to the current sources, rebuilds the corresponding tool and tests it. Any regression with respect to the reference built is sent back to the developer, who is in charge of understanding it before proceeding with commit.
- After they are committed, changes are peer-reviewed by other employees, and specifically senior software developers. They can suggest architectural changes, neat-pick typos, or even find bugs on architectures where the patch was not mailserved.
- All products are built on a daily basis on all the supported platforms, and automatically tested. Regressions are sent back to developers by e-mail, allowing them to fix the regressions. This is a very important step which periodically makes sure that the above steps went fine. It is also an important peculiarity of AdaCore, one of the rare companies to provide such development versions to customers so that they are not blocked by issues in a given release.

As one can see, AdaCore has a significant software infrastructure in place to package sources, build binary packages, test them thoroughly and make them available to customers. This infrastructure is distributed between the main locations of the company; it is truly multiplatform, using 50 different machines on operating systems as different as proprietary unices, GNU/Linux, Microsoft Windows, or Open VMS ; it also includes support for cross environments like WRS VxWorks.

This infrastructure is different from existing free software development components. It includes a bug tracking facility, like bugzilla. However, it provides features which do not exist with bugzilla, like the tracking of ticket numbers for inclusion in particular releases. It includes a build facility, like Source Forge. Unlike source forge, this build facility is truly multi-platforms. Like dejaGNU, it provides a comprehensive automated testing facility. Unlike dejaGNU, testing is tightly integrated with tracking, keeping a direct link from customer reports up to fixes in the technology and granting developers, through the « mailserver » facility, an easy way to test their changes on all platforms. Unlike any software available so far, it also includes a web-server specifically designed for giving customer access a convenient access to all the products they are subscribed to. To summarize, the infrastructure integrates all the components required to provide the complete chain of software services.

3. The future of AdaCore procedures and infrastructure

AdaCore's procedures and infrastructure are however bound to change under a number of trends. First, GNAT Pro is supported every year on more platforms. If some operating systems are getting increasingly obsolete, a huge number of new processors or operating systems are on the radar screen and need to be supported, in particular in the world of cross-development. Beyond the increasing number of such platforms, many new tools have been added to the tool-chain in the recent years, requiring more testing for validation and more rigorous processes.

To cope with this, our procedures have to change and more automation has to be put in the loop. Several areas of improvements can already be mentioned:

- The release monitoring dash-board used to keep track of the status of the various binary packages through the release process was handled manually in the past and needs to be automated to face the growing number of packages. In particular, loops in the build-testing process need to be properly supported;
- The sanity checking phase described above has already been partly automated, with a specific sanity checking program added to speed-up the manual work while keeping the spirit of this

kind of testing.

- The procedure used to arrange the binary packages in their download locations is already partly automated and needs to be completely automated.

In addition, AdaCore is considering making this infrastructure available under a free-software licence because it strongly believe that many free software communities with similar constraints, as well as some of its customers, could take advantage of such an integrated system. However, such a move would require extensively investigating the potential users' base to make sure their needs are properly addressed. For instance, the permission model would probably need some relaxation for use by free software communities, and maybe some strengthening for use by defence software contractors.

In order to make sure no aspect is neglected during this generalization process, a study needs to be conducted with potential users. The first step is to identify potential customers in the free software world and among existing AdaCore customers. A questionnaire with a comprehensive list of questions could be submitted to them, asking questions about their own software development processes, which specific questions about their design process, their tracking process, the way the build releases, validate software and make it available to their customers or users. Using those answers, it would be possible to identify areas where the existing infrastructure would need to be improved, and areas where it is already adapted to the needs of developers.

This kind of methodology has been successfully used as part of the QualOSS project to find out the quality models implemented by some prominent free software projects and could probably be extrapolated for the this specification activity as well.

Conclusion

As the share of free software in software industry gets prominent, it is of utmost importance for free software developers to make sure they are able to match the quality level their users are expecting. AdaCore has developed for this purpose a set of procedures and a matching infrastructure that presents a number of particularities. Compared to other software in this area, it is comprehensive and tightly integrates all its components. In addition, it is designed to support a vast number of different and heterogeneous systems. Eventhough this would require a careful study to make sure it is adequate beyond its own context, AdaCore is considering making these tools available so that the software development community can take advantage of this set of good practices and tools.

For more information

<http://www.adacore.com/>

<http://www.qualoss.org/>