APPLYING AGILE PROJECT MANAGEMENT FOR ACCELERATOR CONTROLS SOFTWARE

W. Sliwinski, N. Stapley, CERN, Geneva, Switzerland

Abstract

Developing accelerator controls software is a challenging task requiring not only a thorough knowledge of the different aspects of particle accelerator operations, but also application of good development practices and robust project management tools. Thus, there was a demand for a complete environment for both developing and deploying accelerator controls software, as well as the tools to manage the projects. As an outcome, a versatile development process was formulated, covering the controls software life cycle from the inception phase up to the release and deployment of the deliverables. In addition, a development environment was created providing management tools that: standardize the common infrastructure for all the concerned projects; help to organize work within project teams; ease the process of versioning and releasing; and provide an easy integration of the test procedures and quality assurance reports. Change management and issue tracking are integrated with the development process and supported by the dedicated tools. This approach was successfully applied for all the new controls software for LEIR, SPS, LHC, injection lines and CNGS extraction.

INTRODUCTION

The Controls Group, in the Accelerators and Beams Department (AB-CO) at CERN, is responsible for the controls infrastructure of all the accelerators at CERN and it provides core systems for accelerator operations, e.g.: central timing, front-end systems, equipment settings management, software interlocks and machine protection, alarms, logging services, sequencer for hardware commissioning, access control, among others. Development of the core systems can not be done in separation and a unified infrastructure is necessary in order to streamline all the efforts and to promote reuse of the common components. Therefore a common development environment was established for software development particularly in the Java programming language, aimed at providing all the necessary services for the controls community.

ABOUT THE DEVELOPMENT PROCESS

Controls Community

Users of the controls infrastructure comprise not only of the members of AB-CO group but also operators in the *CERN Control Centre* (CCC) [1], physicists and equipment experts. Since operators are domain specialists they often provide user requirements for the control systems. On the other hand, experienced software engineers, who are members of AB-CO group, develop frameworks, core components and complete systems. Thus, the users' community is diverse, the level of proficiency in programming varies and many people within it work on temporary basis (such as project associates, fellows and technical students) for the Large Hadron Collider (LHC) start up.

Requirements of the Process

Taking into account the needs of the controls community and the specific nature of the controls domain, a list of requirements can be formulated which characterize the properties of the anticipated development process:

- Maximise ease of use by providing a quick comprehension time (particularly significant for temporary members).
- Standardize development environment (source repository; projects' structure; management of dependencies and 3rd party libraries; build, release and deployment management).
- Automate repetitive tasks (build and release management; running of code coverage and test suites).
- Encourage high quality (code quality; design and code reviews; documentation; test suites).
- Provide complete software configuration management solution (code identification; configuration control; components' consistency review; build management; defect tracking).
- Ensure reproduction of the same artefacts at any time in the future.
- Provide obvious value so teams easily realize its benefits.

Agile Practices for Controls

When developing controls systems at CERN, there is a significant dependence on face-to-face communication with the operators, nearby in the CCC. The work is distributed among several small teams (between 2 and 5 people) often composed of co-located staff and temporary members.

Each team is assigned to one or more projects which are managed by a project leader, who is an experienced software engineer and maintains close collaboration with operators, who provide requirements and domain expertise. User requirements can often change and the resulting features are needed quickly, which in turn leads to the necessity of frequent product releases. Operators are open to test the new releases especially if they know

that their new features are available, thus enabling further feedback for following versions.

Having gained operational experience in the controls domain and having tried several available approaches to project management in the past, including Rational Unified Process (RUP) [2], the agile approach appears to be very promising in assisting teams in developing software in the manner mentioned above. It was perceived that agile methods provide the most benefits for our environment, avoiding the overhead related to heavy-weight processes, where software releases are performed only when significant amount of changes or new features are implemented. Agile practices are not in fact a traditional methodology – it is a collection of guidelines and conventions based on experience, knowledge and tools. Moreover, the approach is based on reflection and continuous improvement of the current environment, responding to changes in technology, development best practices and user requests. Users are not constrained by the process but instead can concentrate more on the development related to their domain.

A complete environment for Java development was provided and only standard Java community tools were used if possible. However in order to address some specific requirements of the controls community two custom solutions were developed: Common Build (build system) and Release (release and version management system) [3] tools.

DEVELOPMENT TOOLS

What follows is an overview of the tools, which grew over the time and they were setup in order to help manage the development. Some of the mentioned tools we are still learning how to exploit fully; others are quite mature in their use.

Versioned Source Repository

All software products are stored in a dedicated repository for accelerator controls software in the CERN central Concurrent Versions System (CVS) [4]. Each product has a well defined directory structure following naming conventions which reflect part of the domain that this product is responsible for.

Common Build

Common Build is a build system, based on Apache Ant [5], providing the common tasks required for development. When the work on new CERN controls system began in 2002, there was a need for a build tool which would help to set up an easy and uniform process that could be applied to new Java development. Due to the lack of a mature build tool available at that time, it was decided to create a custom solution based on standard tools. Common Build provides the following set of common tasks: dependencies management including 3rd party libraries; compilation and packaging; unit testing with JUnit [6]; code quality inspection (using the PMD [7] tool); code coverage reports (using Cenqua Clover [8]); source code and documentation generation; integration with the release system. All products must follow the same directory structure and use common configuration files, thus for the developer, the effort required when working with a project is greatly minimized.

Release

The release process is handled by a separate tool – Release, which is also based on Apache Ant and supports Java, C and C++. The main responsibilities of the Release tool include: ensuring completeness and consistency among components; managing the build process and tools used for builds; ensuring adherence to the organization's development process; managing the software distribution repository; making sure every defect has traceability back to the source. Release tool is composed of a client and server, where the client is fully integrated with the Common Build, but it can be also used standalone. The Release tool extracts product sources from the CVS to a new version directory in the distribution repository and builds the product by calling build tool (e.g. Common Build for Java applications). Subsequently, the new version is installed in a multi-versioned repository without modifying any previous versions.

Continuous Integration

Projects are increasingly leveraging code from other projects. As these "common components" are maintained, any update must be tested with all its dependent projects. A Continuous Integration (CI) server was introduced to automate this increasingly time-consuming task. Using web-based Atlassian Bamboo [9] tool, projects can be marked as dependents of a component whereby they are re-built and their test suite re-run for verification. This minimizes the risk of compatibility problems after a new version is released and reduced the time spent by developers on this task.

The CI server also produces reports based on a project's tests, code coverage, code metrics and inspection. This provides an open account of the health of a project's code base, highlighting areas for improvement. More recently, the acceptance level of unit tests or of some agreed code metric rules (for example, each class file must have at least one comment) can be set. In the future, a project will be considered complete, not only when it provides all the required artefacts, but also when it meets the quality standards.

Issue Tracking

According to the "lean" side of Agile, work waiting to be done should be kept to a minimum. Task switching and keeping work "in process" is a "waste". Ideally, the list of bugs and feature requests waiting for completion should be limited. Nevertheless users and developers still need a

tool to hold their bug reports, feature requests, and project to-do lists, and for this Atlassian Jira [10] is used. Projects can have a large number of stakeholders, so the web interface enables transparency between them showing what work done, work ongoing, work planned and priorities.

Wikis for Collaborative Online Documentation

Project documentation was traditionally created as HTML. Using a wiki has made it easier to create and update reference manuals and user tutorials, with the ability to version pages and easily link between them. Users are also encouraged to participate and add comments to share their experiences or improve the original material. One lesson learned is that, for high quality information, it is still necessary to have a process for reviewing information regularly with a vision of the site content and structure.

Source Code Searching and Inspection

Open inspection and searching of code within CVS is not easy to do with a large code base, which should be a valuable team resource. Although tools like ViewVC [11] are available, the information provided is very limited. Cenqua Fisheye [12], a code repository search tool, has proven to be very useful in this area. Developers can track changes as change sets, search code for duplication, or find how others have tackled a similar problem or used a certain library.

APPLYING THE PROCESS TO PROJECTS

Two example projects which employ an agile development process are LSA [13] and LASER [14].

The LHC Alarm Service (LASER) is an alarm and notification system. It has to be flexible due to its' large number of disparate users including LHC and the whole accelerator chain as well as the technical infrastructure.

The LHC Software Architecture (LSA) project provides homogenous application software to operate the accelerators. It was already successfully used from 2005 onwards to operate the Low Energy Ion Ring (LEIR), SPS and its transfer lines, and LHC, replacing the existing old software.

Both projects tried and benefited from a number of agile practices. They produced a simple, limited feature, working version early on and extended it incrementally. Developers on the projects work within the same room or corridor to lower the barrier of communication. They aim to integrate features into production frequently rather than have multi-featured long-awaited releases. This is particularly important for system stability as it is easier to test and release many smaller versions with relatively few changes, than a major one.

The provided tools are used for everyday development and project management – an interesting example of this is for new feature requests. These are tracked in Jira and presented at planning meetings so users can decide on, and prioritises the backlog list. It is then obvious to all concerned what is to be worked on for the next iteration.

CONCLUSION

In general, agile practices have improved project development in AB-CO group. As a part of this overall environment, the tools mentioned above have helped to reduce development time and organise work in the projects in a standard manner. Some agile elements were found to be in place already such as good communication between development teams and operators. Other elements like continuous improvement still require further refinement. Finally, core tools like Common Build and Release are already acknowledged to be indispensable and are heavily used by all the projects.

REFERENCES

[1] D. Manglunki and P. Charrue, "The CERN Control Centre: Setting Standards for the XXIst Century", ICALEPCS'2007, Knoxville, October 2007.
[2] IBM Rational Unified Process: www.ibm.com/software/awdtools/rup/
[3] G. Kruk et al., "Development process of accelerator controls software", ICALEPCS 2005, Geneva, October 2005.
[4] Concurrent Versions System: http://www.nongnu.org/cvs/
[5] Apache Ant: http://ant.apache.org/
[6] JUnit: http://www.junit.org/
[7] PMD: http://pmd.sourceforge.net/
[8] Cenqua Clover: http://www.cenqua.com/clover/
[9] Atlassian Bamboo: http://www.atlassian.com/software/bamboo/
[10] Atlassian Jira: http://www.atlassian.com/software/jira/
[11] ViewVC: http://www.viewvc.org/
[12] Cenqua Fisheye: http://www.cenqua.com/fisheye/
[13] G. Kruk et al., "LHC Software Architecture [LSA] – Evolution Toward LHC Beam Commissioning", ICALEPCS 2007, Knoxville, October 2007.
[14] Peter Sollander et al., "Alarms Configuration Management", ICALEPCS 2007, Knoxville, October 2007.