MAKING CONTINUOUS INTEGRATION A REALITY FOR CONTROL SYSTEMS ON A LARGE SCALE BASIS

A. Buteau, V. Hardion, S. Le, M. Ounsy, G.Viguier, SOLEIL, Gif-sur-Yvette

Abstract

To support and maintain the control systems of a Synchrotron source, complexity can quickly become difficult to manage due to the large number of software components and different versions deployed. At SOLEIL, it soon became apparent to us that the ability to deploy the same version of software packages across all systems should be a strategic goal. Rigorous development organization and configuration management enable software packages to be built based on release tags put by developers on each software module. Packages integrating all these modules are then built by the continuous integration system *(Maven[1], Hudson[2], CVS[7],)*. These packages are then deployed on around 20 control systems during each machine shutdown, via system administration tools such as *rsync*. To ensure that the continual pace of software changes is acceptable for the people in charge of operating the facilities *(accelerators and beamlines)*, good communication and well organized tests are very important to cope with software regressions or local incompatibilities. Our conclusion will give feedback from over two years' use of this continuous software integration scheme in real operation.

THE CONTEXT

Missions of the ICA Group

Within the SOLEIL Computing Division, the ICA group is in charge of Software for Controls and Data Acquisition for the accelerators and beamlines.

Our mission covers all aspects of the software life cycle:

- Working with machine engineers and beamline scientists to draw up specifications.
- Designing a solution.
- Implementing or subcontracting the development of the identified software modules.
- Maintaining these software component(s) in the long term.
- Providing 24/7 on-call support for over 6000 hours per year.

Relatively Large Scale Deployment

Some twenty different control systems *(one for the accelerators and one per beamline*) are in operation at SOLEIL, each one running on a dedicated TCP/IP network. The various software packages (described below) are hosted on over 100 UNIX servers and 200 CompactPCI crates. Over 100 operator stations (*X terminals or PCs*) provide access to the GUI applications.

Managing a Large Number of Software Components

The layered software architecture of SOLEIL Control Systems encompasses an impressive number of different components:

- Three important frameworks are interconnected:
  - Tango [3]
  - GlobalSCREEN [4]
  - Passerelle [5]
- About 300 DeviceServer classes.
- Over 150 elementary ATK [6] widgets.
- Over 40 GlobalSCREEN applications.
- Several hundred Passerelle "actors" and sequences.

"Extreme Programming" [8] Project Management

Getting specifications from our users is often a difficult task. For this reason an iterative approach is preferred, whenever possible, to provide software versions to our users at an early stage (*to get their feedback and ease further specifications*) and on a regular basis (*even if not all features are complete*). In our experience, this method has proved successful in quickly obtaining user feedback, avoiding over-specifications and minimizing integration problems.

GOLDEN RULE No. 1: WE KNOW WHAT WE DEPLOY

Guidelines

- "**Only known AND identified versions should run on production systems"**: it is the responsibility of each module developer to define the production version with a CVS tag
- "**All binaries must be constructed from source code independently of the development environment**": this ensures that all source code is versioned into the CVS system

Package Organization

Software modules are packaged in four official software distributions which follow our project breakdown

- DEVICE_ROOT: package containing over 300 Tango DeviceServer binaries.
- SOLEIL_ROOT: package containing standard Tango GUI applications.

- GLOBAL_ROOT: package containing the SCADA tool used at SOLEIL and the GUI components provided for GlobalSCREEN application development.
- PASSERELLE_ROOT: package containing the Passerelle sequencer and the process components used at SOLEIL.

One engineer/technician is responsible for the integration of each package.

Integration Tools Used to Build Each package

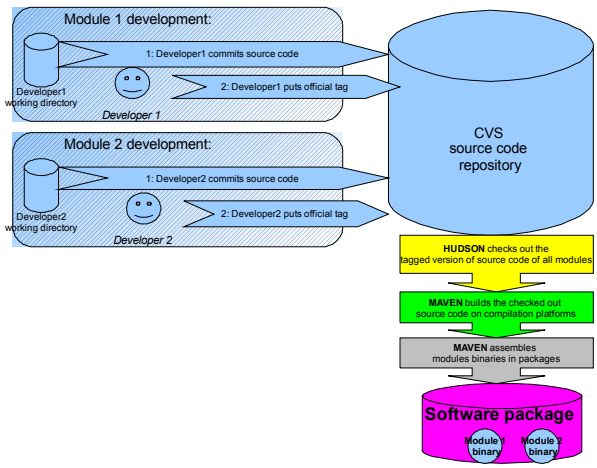

Figure 1: Module build and package integration.

Build Cycle

- To find potential incompatibilities between software layers, each package is built every week: a so called **Snapshot** version is then produced without being deployed.
- During machine shutdown (*preferably at the beginning of the shutdown*), an official version (called **Release**) of the package is built by Maven/Hudson, and is deployed everywhere.

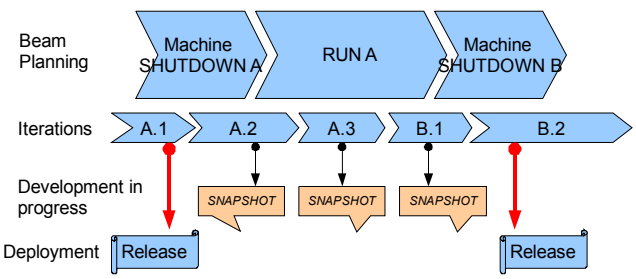

Figure 2: Build planning.

GOLDEN RULE No. 2: WE DEPLOY THE SAME VERSIONS EVERYWHERE

Guidelines

- Official releases of all software packages are installed on the accelerators and all the beamlines without any exceptions.
- If problems arise with new versions, it must always be possible to use the previous package or module version. Nevertheless, old versions will automatically be superseded by new versions at the next shutdown: such exceptions to the rule are therefore limited in time.

Tools Used

- Each package is itself versioned in the CVS system
- The new version of the package is downloaded on all Control Systems using standard Unix tools (*rsync and wget commands*).
- A symbolic link is then modified to point to the new version, which is, from this moment, officially in production.

RULE No. 3: WE DEPLOY FREQUENTLY AND WITH STRONG SUPPORT TO OUR USERS

Guidelines

- The life of a synchrotron facility is punctuated by short accelerator technical shutdown periods (generally lasting less than two weeks).
- We use these periods to install **new versions** of all software packages **on all** control systems
- This obliges software developers to make frequent deliveries, which helps our iterative project management approach.

Tools and Organization

After a new version is installed, non-regression tests are organized on each beamline with a software engineer and a beamline scientist.

Moreover, the first day of beam delivered to the beamlines, one engineer is physically present on the beamline to help correcting or reporting small defects.

So called "patches" (*i.e. minor corrections to the official packages*) may be redistributed to all systems, during this first day of beam (*which is not open to external users*).

Communication with Our Users

Once packages are produced, we communicate to our users a list of the modules which have changed, with an analysis of the level of risk associated with each change. This risk level is determined by the software developer of the module. Using this "risks spreadsheet", functional and non-regression tests can be targeted to the modules which have the riskiest changes.

Table 1: Risks Analysis Spreadsheet

Programme	Projet /Application	Fonction	Evaluation du risque
Sources/Expériences	Mambo	Archivage et application d'extraction des données	FAIBLE
Sources/Expériences	Bensikin	Gestion des snapshots	FAIBLE
Sources/Expériences	machinestatus	Informations sur l'état de la Machine et des têtes de lignes	FAIBLE
Sources/Expériences	charleston	Application de monitoring et traitement des images faisceau	FAIBLE
Sources/Expériences	salsa	Application de scan	MOYEN
Expériences	Tumba	Visualisation des données des détecteurs MCA	NOUVELLE APPLICATION

Moreover, on the Monday before beam is delivered to beamlines, software changes are presented to all beamline and machine groups at SOLEIL's official Operation meeting.

Lastly, special attention is given during this shutdown period to the new problems described in our bug tracker.

THE RESULTS

Good on-call Duty Support

Given the large scale deployment and large number of software components, the probability that the software engineer called on duty is an expert on the failing software or control system is statistically low. Thanks to our two Golden Rules, we can be confident that only tested and official versions are deployed on production systems. Moreover, the way our packages are organized enables us to revert to the previous version simply by changing a symbolic link. Lastly, as versions are the same on all systems, we avoid different behavior from one control system to another.

The number of software support calls from users has decreased dramatically, vindicating our policy of continuous integration.

Table 2: On-call Duty Support Statistics

Evolution of on-call duties			
	2007	**2008**	**2009 *(estimation)***
Number of calls	128	137	152
Control Systems deployed	6	12	22
Beam hours	4900	4600	6000
Tango devices	8000	15000	20000
Quality indicator *Calls*10000/(beam hours*devices)*	**3,3%**	**2,0%**	**1,3%**

Software Quality has Improved

The great majority of software components are used on more than one control system. This facilitates the discovery of new bugs, and Golden Rule No. 2 ensures that corrections are then automatically made available throughout the institute at the next software deployment. As an example, this mechanism has been used to put new major versions of Tango kernel libraries *(including bug fixes and new features)* into production on over 20000 devices during a single shutdown.

Last but not least, new features on a particular device or GUI application are also available to everybody, which avoids duplication of efforts for specifications and tests.

Better Management of Software Integration Risks

Our layered architecture allows us to provide our users with a substantial number of applications providing high level services and features for equipment controls, data acquisition, supervision and process control. However, integrating all layers (*the majority of which are not developed at SOLEIL)* is a technical challenge. Thanks to the continuous integration process, software compatibility problems between layers are discovered before the deployment phase. Once new versions are deployed, our functional test organization allows us to discover more subtle integration problems.

CONCLUSION

It was initially very difficult to convince accelerator and beamline control system users that changing software versions all the time would be beneficial for everybody in the long term. Their natural reaction was: *"Why do you want to change the software on my beamline? I didn't encounter many problems during the last run, and I don't want to spend time retesting everything*".

We then had to justify that on call duty support would quickly become impossible if versions were different everywhere. Nevertheless, our strong commitment to quickly correct and manage integration problems during new version deployment phases has been essential in ensuring the acceptance of continuous integration at SOLEIL. We can even say that users have come to appreciate the practice, which demonstrably contributes to software quality enhancement and helps in defining project milestones during the development phase.

Last but not least, the majority of accelerator and beamline managers are now convinced that it is probably the best way to manage changes and risks, in a synchrotron world which is permanently moving and evolving.

REFERENCES

[1] Hudson: https://hudson.dev.java.net.
[2] Maven: http://maven.apache.org.
[3] Tango: http://www.tango-controls.org.
[4] K. Saintin, V. Hardion, M. Ounsy, "How to Use a SCADA for High-Level Application Development on a Large-Scale Basis in a Scientific Environment", ICALEPS'07, Knoxville, Tennessee - USA, Oct 2007.
[5] A. Buteau, M. Ounsy, G. Abeille "A Graphical Sequencer for SOLEIL Beamline Acquisitions", ICALEPS'07, Knoxville, Tennessee - USA, Oct 2007.
[6] F. Poncet , J.L. Pons, "Tango Application Toolkit", ICALEPS'05, Geneva, Oct 2005.
[7] http://en.wikipedia.org/wiki/Concurrent_versions_system.
[8] http://en.wikipedia.org/wiki/Extreme_Programming.