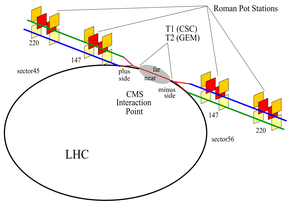
Regression testing in the TOTEM DCS

F. Lucas Rodríguez et al.
on behalf of the TOTEM collaboration

BIG.BROTHER — TOTEM DCS - Detector Control System

Introduction

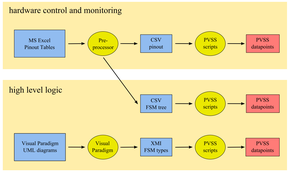
- The TOTEM (total and elastic measurements) experiment at CERN [1, 2] measures the size of the proton and also monitors accurately the LHC's luminosity. To do this TOTEM must be able to detect particles produced very close to the LHC beams.
- TOTEM consists of 'Roman Pot Stations' (RP), 'Cathode Strip Chambers' (CSC) Telescope 1 (T1) and 'Gas Electron Multipliers' (GEM) Telescope 2 (T2). The T1 and T2 detectors are located on each side of the CMS interaction point in the very forward region, but still within the CMS cavern. Two Roman Pot stations are located on each side of the interaction point at 220 m and 147 m inside the LHC tunnel. Each Roman Pot station consists of two groups of three Roman Pots separated by a few meters.



- The TOTEM Detector Control System (DCS) team developed a set of automation scripts and a C# tool to generate those datapoints in a reliable and unified way [3]. This makes it possible to reconfigure of the whole system within hours (within one working day) and a homogenizes the behaviour among subsystems.

Motivation

- The purpose of our novel regression tool is to provide an easy way to evaluate the consistency of the final project with an earlier reference project which is considered to be working correctly.
- The custom code for the automated generation of datapoints is very reliable, however there is always need for new features and minor changes to deal with ongoing project maintenance. Those modifications somehow have undesirable **side-effects** in other detectors or parts of the code. The TOTEM DCS group was faced with a software quality assurance issue.
- PVSS has the tool `pvss00ascii.exe` to export all datapoint configuration into a plain text proprietary format. The resulting file has the extension `.dpl`. However many times these files include too much information such as deployment system hardware status warnings. Those `.dpl` files cannot even be directly compared between versions, as all the timestamps change. Using PVSS `.dpl` as an exchange mechanism is not optimal and can lead to many problems.



- The sources of information are MS Excel files (with thousands of rows) and a Visual Paradigm project. A custom preprocessor written in C#, is used to parse the MS excel files applying some heuristics to generate the PVSS datapoints names and the corresponding aliases. Then based on the aliases it generates a tree structure following the Product Breakdown Structure (PBS) of the detector. However those inputs cannot be easily compared with a `diff` tool either. The TOTEM DCS has a centralized definition for all the archiving, alarms, units,... configuration. A dedicated procedure enforces that all the datapoints match certain filters and conform to central configuration standards.
- The procedure that the regression tool uses for comparing system is based on building a new PVSS project from scratch, and installing all the JCOP and TOTEM components.

Previous works

- In parallel with our development, the JCOP framework team (in the EN-ICE group at CERN) also has a Quality Assurance effort. They also run automated tests against PVSS, both code and the User Interface.
- However our aim is slightly different: we want to validate the integrity of our final system against our changes. We do not try to commission PVSS, the JCOP framework nor the Operating System. We consider that all of these software layers are reliable and will not introduce problems for our tests.
- We focus our development effort in having a reliable back-end and accept that our User Interface can have minor problems that users will notify to the development team.

Conclusion

- We could easily adapt to a continuous running scenario by extracting `.dpl` files from the production systems and verifying using a system generated in a development environment. Regression testing and comparison applies not only across different releases but also among environments (production and development).
- The tool itself does not provide full code coverage testing and has many limitations. However if after introducing changes, the `.dpl` file evolves as expected and there are no additional error messages the developers are extremely confident that an upgrade will not produce any side effects, (or at least it will not be any worse that it was before!).
- If a JCOP component alone is changed without changing our TOTEM components, we can even identify bugs in the JCOP framework. We are able to explore the compatibility with newer component versions before the production systems are upgraded. We could also validate different PVSS versions and PVSS patches using this the same procedure.

References

[1] G. Anelli, G. Antchev, P. Aspell, *et al.*, 'The TOTEM experiment at the CERN Large Hadron Collider,' *Journal of Instrumentation*, vol. 3, pp. 1–113, Aug. 2008, ISSN: 1748-0221. DOI: 10.1088/1748-0221/3/08/S08007. [Online]. Available: http://iopscience.iop.org/1748-0221/3/08/S08007.

[2] G. Antchev, P. Aspell, I. Atanassov, *et al.*, 'Proton-proton elastic scattering at the lhc energy of sqrt(s) = 7 tev,' *A letters journal exploring the frontiers of physics*, vol. 95, Aug. 2011, ISSN: 1286-4854. DOI: 10.1209/0295-5075/95/41001. [Online]. Available: http://iopscience.iop.org/0295-5075/95/4/41001.

[3] I. Atanassov, F. Lucas Rodríguez, P. Palazzi, *et al.*, 'Automation tools in the software development of the totem detector control system,' in *2010 IEEE Nuclear Science Symposium Conference Record*, K.-P. Ziock, Ed., Oct. 30–Nov. 6, 2010, pp. 327–332, ISBN: 978-1-4244-9106-3. DOI: 10.1109/NSSMIC.2010.5873774. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5873774, (On behalf of the TOTEM Collaboration).

Architecture

Formalization of the steps

- Kill any PVSS related process running in the machine.
- Export all the components stored in SVN repositories as defined in the config files.
- Export from SVN the reference `.dpl` file.
- Configure the Windows firewall for all the PVSS projects.
- Create a PVSS project. We invoke the standard PVSS procedure from an special User Panel not linked to any particular project. Currently this panel is stored in the component `totInstallation`.
- Modify the generated project to include the `fwInstallation` and `totInstallation` in the `proj_path` variable of its `config` file.
- Modify the generated project `config` file to include all the components specified in the config file (and exported from SVN) into a `totInstallation` section.
- Start the project.
- Append and additional manager from `totInstallation` into the project using the PMON TCP protocol.
- Wait that `totInstallation` has finished installing all the desired components. The tool monitors the custom manager status and the global project status using the PMON TCP protocol. There can be several restarts of the project during this process.
- Stop the project using the PMON TCP protocol.
- Merge PVSS logs and provide a resume of the errors during the execution; save them to the hard disk.
- Export the datapoints into a `.dpl` file.
- Commits the `.dpl` file into SVN.
- Clean up the resulting and reference `.dpl` files.
- Compare both `.dpl` files and generate a temporary report in the hard disk.
- Send by email the report and PVSS logs.

Details on the project config file manipulations

- The project `config` file is updated before the start up of the project. The first step is to introduce new search paths:
- Later in the file we list all the desired components and where to find them. This file also accepts additional sections identified by `[sectionname]` followed by several entries of the form `key=value`.

totInstallation

- A key step for the process is a panel able to create a project just executing a command line:
- Additionally `totInstallation` reads all the entries in the `[totem]` section of the project `config` file.
- It verifies if the components are installed, and forces their installation one by one, making sure that the `postInstall` scripts are properly executed and finished before advancing to the next one.

Cleanup of .dpl files: The castrator

- Castration of the datapoint list means to preserve only the structural information of these datapoints and not their value contents.
- Each distinct section of the datapoint file has to be inspected to determine the right columns to be copied.

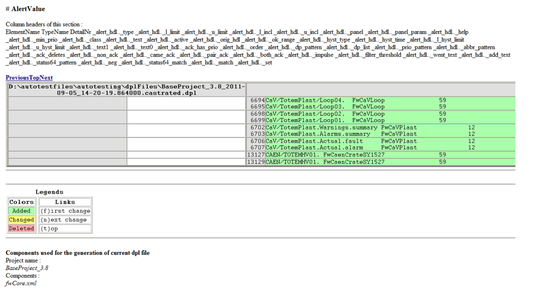
```
<castrateConfig>
    <section index="ElementName" name="DpValue">
        <filterIgnore>
          <match regexp="^_Stat_event_\d\..*"/>
          <match regexp="^_DistManager\..*"/>
          <match regexp="^_DistConnections\..*"/>
          <match regexp="^_mp_COUNTER1\..*"/>
        </filterIgnore>
        <columnsToCopy>
           <!--The index 'ElementName' is copied as well-->
            <column name="TypeName"/>
            <!--<column name="_original.._value"/>-->
            <!--<column name="_original.._stime"/>-->
        </columnsToCopy>
    </section>
</castrateConfig>
```

PVSS logs filtering

- Only the SEVERE, FATAL and WARNING error messages that not explicitly suppressed are included in the HTML body of the final report.

Reporting

- Changes between the `.dpl` files are compared using the Python `difflib` libraries and included in the report.
- On the left side it shows the validated file, and on the right side any modifications. Python `difflib` uses a color encoding to clarify if they are additions, removals, or changes.
- At the end of the report there is a list with the components used, the filtered error messages, and information about the running time and OS platform used for the test. The full PVSS log of the installation process is emailed as a compressed attachment.



Possible improvements

- Do a comparison not only of `.dpl` files, but also of the directory structure.
- Partial extraction of `.dpl` files and their comparison could help in unit testing.
- Extend the regression testing to cover User Interface aspects.
- Provide better feedback about the time used for every component installation.
- The Python comparison library uses too much memory (>2 GB).
- Monitor CPU usage.
- Mailing lists do not accept mails greater than 10 MB.
- Improve the JCOP `fwInstallation` tool to have a local back-end of desired PVSS components for the project, and merge it with `totInstallation`.