PLC Factory: Automating routine tasks in large-scale PLC software development

G. Ulm, F. Bellorini, D. Brodrick, R. Fernandes,
N. Levchenko, D. Piso Fernandez

EUROPEAN SPALLATION SOURCE

Problem

The European Spallation Source ERIC (ESS) in Lund, Sweden, is building large-scale infrastructure that is projected to include hundreds of programmable logic controllers (PLCs). The problem is:

1. PLCs directly control hardware, thus programming errors can have serious consequences
2. Programming PLCs is repetitive and error-prone
3. Some repetitions are not trivial to automate

Contributions

PLC Factory is an application for automating repetitive tasks associated with PLC programming. It relies on an in-house configuration database, CCDB, which stores information for each device instance and device type. PLC Factory is a template-based substitution engine that performs the following tasks:

1. direct substitution, i.e. for a given device d, use property p as specified in the corresponding CCDB entry for d
2. identification of shared properties between devices, in order to remove redundancies in CCDB
3. automatic counters management for specifying PLC memory address offsets in EPICS database records

Results

The time complexity of PLC Factory is $O(n)$, where n the number of devices. On a 2011 MacBook Air (1.8 GHz), PLC Factory processes three different template IDs for a tree with 40 devices in 7 seconds. Manually creating those files would take many hours.

References

[1] Borrowman, A. J., & Taylor, P. (2016, July). Can your software engineer program your PLC?. In SPIE Astronomical Telescopes+ Instrumentation (pp. 99131S-99131S). International Society for Optics and Photonics.

[2] Casas-Cubillos, J., Gomes, P., Gayet, P., Varas, F. J., Sicard, C. H., & Pezzetti, M. (2002). Application of object-based industrial controls for cryogenics (No. CERN-LHC-2002-007-IAS).

[3] Cockrell, L., & Sander, T. M. (1992). Selecting a man/machine interface for a PLC-based process control system. IEEE transactions on industry applications, 28(4), 945-953.

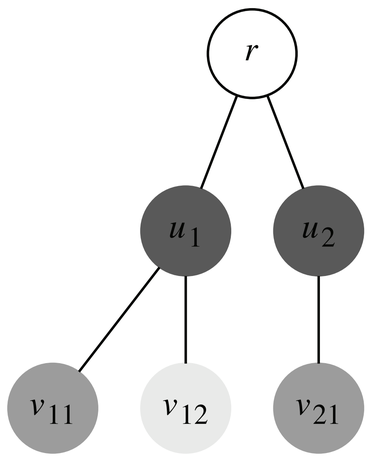
[4] Dalesio, L. R., Kraimer, M. R., & Kozubal, A. J. (1991, November). EPICS architecture. In ICALEPCS (Vol. 91, pp. 92-15).

[5] Zaharieva, Z., Peryt, M., & Martin Marquez, M. (2011, October). Database foundation for the configuration management of the CERN accelerator controls systems. In Conf. Proc. (Vol. 111010, No. CERN-ATS-2011-206, p. MOMAU004).

Solution

The substitutions outlined in *Contributions* remove most if not all of the repetitions of large-scale PLC software development. We highlight four aspects of PLC Factory.

Dependency Trees CCDB describes dependency relationships between devices; a *dependency tree* explicitly models those. In the example below, r is the root device and controls the devices u_1 and u_2, of which the former controls v_{11} and v_{12}, and the latter v_{21}. Trees can be arbitrarily deep.



Template Files Template files are text files with a fixed structure. PLC Factory consumes template files for creating EPICS database records files and SCL code blocks for TIA Portal. Device types may have template files with particular IDs attached to it in CCDB. PLC Factory *dynamically* replaces fields within a template file. A simplified example is the substitution of a field `DEVICE_NAME` by the concrete name of a device that is an instance of the device type this template is associated with.

Substitution Engine Pseudo-code of the core of the substitution engine is given below. The operator $\oplus$ is a shorthand for processing template files. It is applied to a device instance x and a specific template. Templates are retrieved by a function t that takes as its input a template ID and the device type of x, which is determined by the function d applied to device x. Thus, the resulting operation is $x \oplus t(id, d(x))$.

In addition, we define the header file $h(id, d(r))$ as well as the footer file $f(id, d(r))$.

Data: CCDB, root device r, template ID id
Result: list *out* containing text for post-processing
begin
  $out \leftarrow \varnothing$ ▷ collected output
  $ds \leftarrow r$ ▷ list of devices
  while ds not $\varnothing$ **do**
    $d \leftarrow ds$.pop()
    $cs \leftarrow d$.controls() ▷ CCDB lookup
    while cs not $\varnothing$ **do**
      $c \leftarrow cs$.pop()
      if $t(id, d(c)) \in CCDB$ **then**
        $out \leftarrow out + c \oplus t(id, d(c))$
        $cs' \leftarrow c$.controls()
        $ds \leftarrow ds + cs'$
      end
    end
  end
  $out \leftarrow r \oplus h(id, d(r)) + out + r \oplus f(id, d(r))$
end

PLCF$^\sharp$ PLCF$^\sharp$ is an embedded domain-specific language for flexible substitutions. It solves two problems: resolving shared property values and manual memory management. For the former, consider the expression `[PLCF# ^(Offset) + 1]`. Here, PLC Factory looks up the property `Offset` by traversing the device tree upwards. For the latter case, consider the expression `[PLCF# ^(Offset) + Counter1]`. In this example, counter variables automate assigning memory addresses. The user only has to specify which counter should be incremented. A large skeleton file for TIA Portal may contain hundreds of memory locations. PLC Factory completely automates their definition and ensures there are no overlaps.

Future Work

PLC Factory is a command-line application. We intend to turn it into the backbone of a web-based GUI-driven application. Further, we consider extending PLC Factory by adding an exporter to automatically generate operator interfaces (OPIs). PLC Factory can be easily tailored to use other database backends as well, so adding interfaces would be another suitable next step. However, PLC Factory does not rely on domain knowledge. Due to its generic approach to template processing it could be used in many other domains as well, as it is a universal template-based substitution engine.

Source Code

PLC Factory has been written in Python 2.7. The application is used in production and is actively maintained by ESS. We made the source code available under the third version of the GNU General Public License (GNU GPLv3). The project repository is:

`https://bitbucket.org/europeanspallationsource/ics_plc_factory`.