CONTINUOUS INTEGRATION FOR AUTOMATED CODE GENERATION TOOLS

I. Prieto Barreiro, W. Booth, B. Copy. CERN, Geneva, Switzerland

Abstract

The UNICOS (*UNified Industrial COntrol System*) [1] framework was created back in 1998 as a solution to build industry like control systems. The Continuous Process Control package (CPC) [2] is a UNICOS component that provides a methodology and a set of tools to design and implement industrial control applications. UAB (*UNICOS Application Builder*) [3] is the software factory used to develop UNICOS-CPC applications. The constant evolution of the CPC component brought the necessity of creating a new tool to validate the generated applications and to verify that the modifications introduced in the software tools do not create any undesirable effect on the existing control applications. The uab-maven-plugin is a plug-in for the Apache Maven build manager that can be used to trigger the generation of CPC applications and verify the consistency of the generated code. This plug-in can be integrated in continuous integration tools - like Hudson or Jenkins - to create jobs for constant monitoring of changes in the UAB framework that will trigger a new generation of all the applications located in the source code management.

INTRODUCTION

UNICOS (*UNified Industrial COntrol System*) is a CERN (*European Organization for Nuclear Research*) framework designed to develop industrial control system applications. It provides a methodology, several object libraries and a set of tools to generate the control code for these applications. The framework provides several components used in different projects, such as CPC (*Continuous Process Control*), CIET (*Cryogenics Instrumentation Expert Tool*), QPS (*Quench Protection System*) and SURVEY (Control system for aligning the LHC magnets) [4]. These components target very different development platforms like Industrial PLCs (*Programmable Logic Controllers*), FESA (*Front-End Software Architecture*) front-end computers or SCADA (*Supervisory Control And Data Acquisition*) packages.

UAB (*UNICOS Application Builder*) is a modular software tool developed using Java and Jython programming languages and designed to generate either the control code or the configuration files for the different UNICOS applications. Like the UNICOS framework itself, UAB is made up of several components (CPC, CIET, SURVEY, ...) to allow the generation of the different types of applications. Each UAB component consists of one or more software modules (Figure 1). The UAB software modules are classified in three main categories: core, generation plug-ins and resources packages [2].

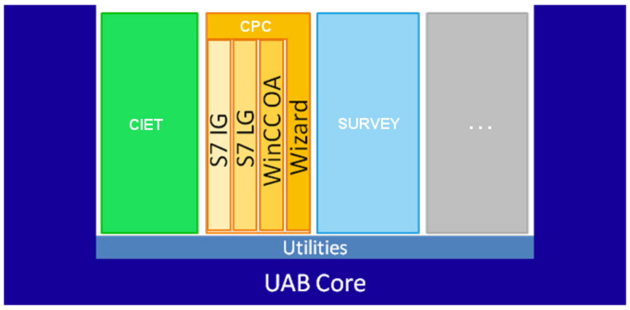


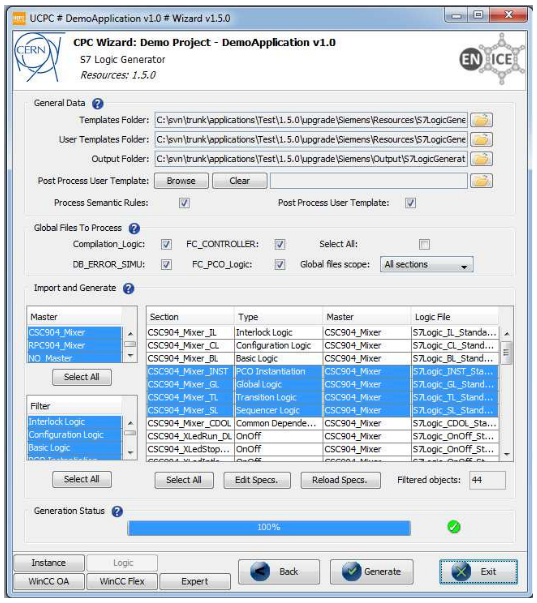
Figure 1: UAB architecture overview.

This software modularity brings several benefits with respect to the continuous improvement of the software:

- The modules can be reused by other modules improving the software quality by avoiding code duplication and reducing the maintenance effort.
- Each module can be implemented by different teams with different knowledge domains (e.g. Software Engineers, Control Engineers, etc.).
- Each module can have an independent life-cycle.

AUTOMATED GENERATION TOOLS

UAB provides a set of wizards with a user-friendly interface that are helpful during the creation, configuration and development of the control applications (Figure 2).



UCPC # DemoApplication v1.0 # Wizard v1.5.0

CPC Wizard: Demo Project - DemoApplication v1.0
S7 Logic Generator
Resources: 1.5.0

General Data
Templates Folder: C:\svn\trunk\applications\Test\1.5.0\upgrade\Siemens\Resources\S7LogicGene
User Templates Folder: C:\svn\trunk\applications\Test\1.5.0\upgrade\Siemens\Resources\S7LogicGene
Output Folder: C:\svn\trunk\applications\Test\1.5.0\upgrade\Siemens\Output\S7LogicGenerat
Post Process User Template: Browse Clear
Process Semantic Rules: ✓ Post Process User Template: ✓

Global Files To Process
Compilation_Logic: ✓ FC_CONTROLLER: ✓ Select All: ☐
DB_ERROR_SIMU: ✓ FC_PCO_Logic: ✓ Global files scope: All sections

Import and Generate

Master
CSC904_Mixer
RPC904_Mixer
NO_Master

Filter
Interlock Logic
Configuration Logic
Basic Logic

Section	Type	Master	Logic File
CSC904_Mixer_IL	Interlock Logic	CSC904_Mixer	S7Logic_IL_Standa...
CSC904_Mixer_CL	Configuration Logic	CSC904_Mixer	S7Logic_CL_Stand...
CSC904_Mixer_BL	Basic Logic	CSC904_Mixer	S7Logic_BL_Stand...
CSC904_Mixer_INST	PCO Instantiation	CSC904_Mixer	S7Logic_INST_Sta...
CSC904_Mixer_GL	Global Logic	CSC904_Mixer	S7Logic_GL_Stand...
CSC904_Mixer_TL	Transition Logic	CSC904_Mixer	S7Logic_TL_Stand...
CSC904_Mixer_SL	Sequencer Logic	CSC904_Mixer	S7Logic_SL_Stand...
CSC904_Mixer_CDOL	Common Depende...	CSC904_Mixer	S7Logic_CDOL_Sta...
CSC904_XLedRun_DL	OnOff	CSC904_Mixer	S7Logic_OnOff_St...
CSC904_XLedStop...	OnOff	CSC904_Mixer	S7Logic_OnOff_St...

Filtered objects: 44

Generation Status
100%

Figure 2: Wizard panel to generate code for Step 7.

ISBN 978-3-95450-139-7

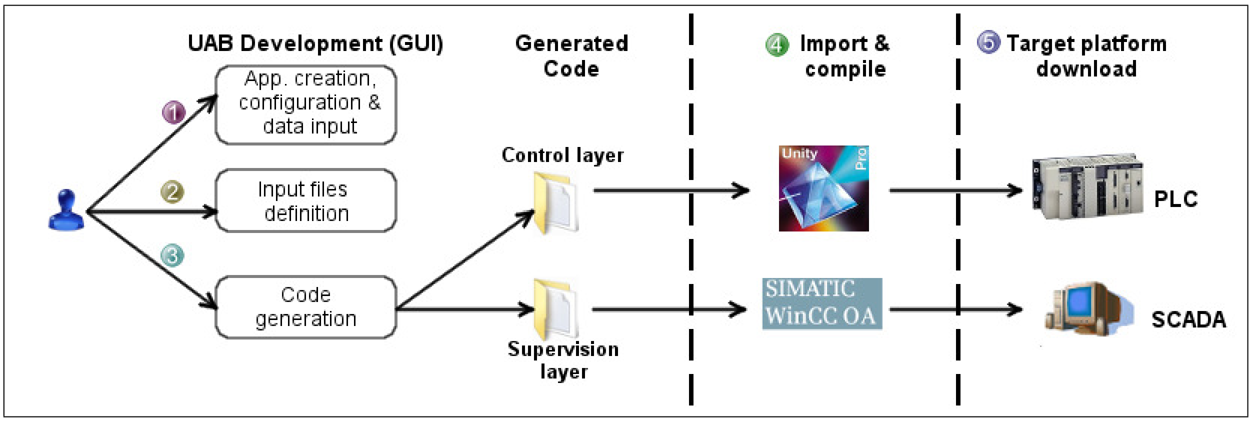


Figure 3: UAB application development workflow.

To develop UNICOS applications using UAB, it is necessary to select the appropriate wizard depending on the application package (CIET, CPC, SURVEY, ...), the target platform for the application (e.g. Siemens or Schneider in the case of PLC-based systems), define the necessary input files (e.g. device specification list) and introduce the required parameters to configure the system (e.g. the application name and version, communication parameters, etc.). Once the application configuration is completed, it is possible to trigger the execution of the various plug-ins to generate the code to be imported in the target platform software (Simatic Step 7, Unity, WinCC OA, ...) (Figure 3).

This development workflow, based on GUIs, is suitable to create new applications from scratch but it is tedious when the developer is generating large or multiple applications. In these cases the generation procedure is a repetitive, error-prone and time consuming task.

To tackle these cases UAB offers a different approach to generate applications reducing the interaction with the developer. The uab-maven-plugin is a plug-in for Apache Maven that can be used to trigger the generation of UNICOS applications without further user intervention.

Apache Maven

"Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information." [5]

Automated Generation with Maven

To perform the generation of UNICOS applications using Maven, it is necessary to create a build XML (*eXtensible Markup Language*) file known as the project object model (POM). This file contains all the necessary information about the Maven project as well as the Maven plug-in configurations to be used during the *build process* (or *generation process* in the current context).

The minimum contents required in the POM file to perform generations of UNICOS applications are:

- Maven coordinates: three elements (groupId, artifactId, version) used to identify a specific version of the Maven project.
- Dependencies definition: it defines the software dependencies required to *build* the Maven project (or, in the current context, to *generate* the UAB applications), i.e. the version of the uab-maven-plugin and the UAB plug-ins that will be used to perform the generation.
- Executions of the uab-maven-plugin: the executions are used to define the location of the UAB applications to generate, the list of UAB plug-ins that will be executed and to modify any of the application parameters before the generation (like the plug-in output folders, enable/disable the execution of the semantic rules or the date format for the log files). It is possible to define several executions to group applications of different nature, like Siemens CPC applications, Schneider CPC applications, CIET applications, etc.

In the case of CPC applications it is possible to go one step further and invoke the compilers of the target platforms (Step 7 for Siemens or Unity for Schneider) after a successful code generation. This compilation will verify whether the generated code fits in the available memory and if it is correct syntactically (e.g. the control structures are correct, the parenthesis are balanced) and semantically (e.g. the variables used in the PLC code are declared, the functions invoked exist).

Additionally, it is also possible to include the configuration of any other existing maven plug-ins to achieve fully automated generations, for example:

- maven-scm-plugin: it is possible to define the application location in the source code management to perform a checkout of all the source files of the application before executing the code generation.
- maven-diff-plugin: used to find the differences between the generated files with a previous version. This is useful to track the changes on the generated files after applying any modifications in any of the input files of the application.

Once the POM definition is completed it is possible to start the generation of the defined applications using a simple maven command: *mvn compile*.

In addition to the UNICOS application development, the uab-maven-plugin is very useful during the software development phase of the different UAB modules. For example, after a source code refactoring or after the development of new features in any software module, it can be used to verify that the generated outputs are equivalent or the modifications in the module did not introduce any undesirable side effect.

The next step is to use continuous integration tools to automatically trigger the generation of the applications so the developers can focus on the application development itself rather than the generation procedure.

CONTINUOUS INTEGRATION

”Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.” [6]

With the help of continuous integration tools - like Hudson or Jenkins - it is possible to create jobs that constantly monitor changes in the source code management (e.g. Apache Subversion, Git) and trigger a generation of the related applications to verify that the modifications do not break the generation. The jobs can be configured to send e-mails to the individuals who broke the build in the case of a failure during the generation of the applications.

The tools used to structure the build automation are (Figure 4):

- Apache Subversion: used as sofware versioning and revision control system.
- Apache Maven: software management tool used to define standard builds and to produce structured build results.
- Hudson: continuous integration tool used for build automation and software deployment.
- Sonatype Nexus: software repository manager used for dependency resolution and as storage location for the deployed software.
- Atlassian Confluence: team collaboration software used to deploy the system’s online documentation and to provide the links to download the deployed software.

With the help of these tools, it is also possible to automate the software release procedure, thereby avoiding repetitive tasks such as tagging the source code, generating the API documentation, packaging the necessary files in a distributable format, uploading the packaged files to a server or repository manager or updating the project’s online documentation. Once the Maven project and the Hudson job are properly configured all those tasks can be completed in two simple steps: introduce the release parameters and trigger the release procedure using the user-friendly interface provided by Hudson (Figure 5).

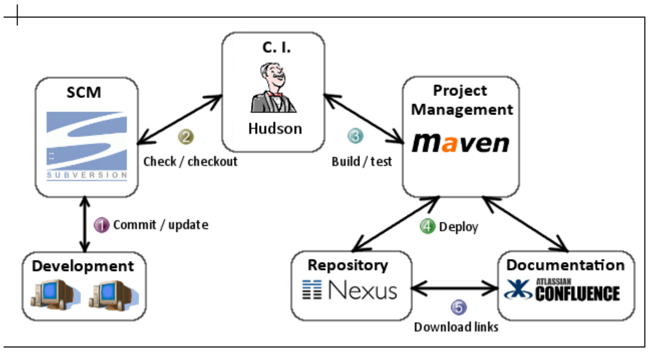
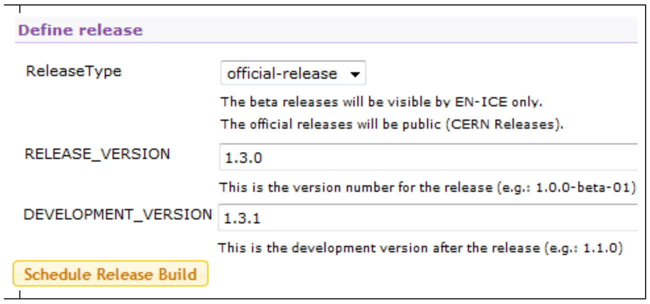


Figure 4: Build automation structure.


Define release

ReleaseType	official-release The beta releases will be visible by EN-ICE only. The official releases will be public (CERN Releases).
RELEASE_VERSION	1.3.0 This is the version number for the release (e.g.: 1.0.0-beta-01)
DEVELOPMENT_VERSION	1.3.1 This is the development version after the release (e.g.: 1.1.0)

Figure 5: Parametrized release definition in Hudson.

FUTURE DEVELOPMENT

The tools and procedures described allow a high level of automation for the generation of control system applications - improving their overall quality - but there is still room for improvement in several areas.

Generation Output Unit Testing

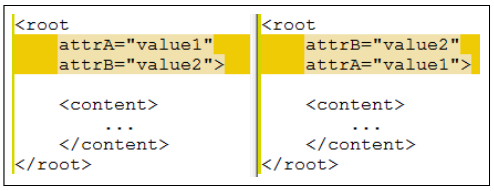
While our current UAB integration testing approach relies on the comparison of output files to expected contents, such approach does not allow the formulation of sophisticated assertions, nor does it allow to easily pinpoint the cause of a difference in test results, like one would expect from a unit testing framework such as JUnit [7].

This stems from the fact that the UAB generation output cannot currently be parsed, for instance PLC code cannot yet be turned into an abstract syntax tree. Having such a capabilities would allow to express fine-grained verification assertions that would greatly enhance the precision of our unit tests and locate the cause of a regression in the UAB codebase.

ISBN 978-3-95450-139-7

Generated Code Comparison

Currently, the comparison of the generated outputs of an application is made using text comparison tools like the maven-diff-plugin. This approach is appropriate when the format of the generated files is plain text but it becomes unusable for other formats like XML. For example, when an XML content is serialized, the order of the element's attributes is not significant and can not be enforced. As a result, different generations can create equivalent XML files but not identical from a text comparison point of view, which makes the file comparison more complex (Figure 6).



```
<root                        <root
    attrA="value1"               attrB="value2"
    attrB="value2">              attrA="value1">

    <content>                    <content>
       ...                          ...
    </content>                   </content>
</root>                      </root>
```

Figure 6: XML comparison using diff tools.

Assist the UAB Developers

One of the challenges that the developers face during the first development phases of a new UAB component is that, at least, one UAB plug-in and a resources package must exist from the beginning (the plug-in input files are defined in the resources package and the code generation is driven by the plug-in using the scripts provided by the resources package). To facilitate this task, the functionality of the uab-maven-plugin is being extended to:

- Generate the skeleton of the resources package, plug-ins and wizards.
- Add new UNICOS objects to an existing resources package.
- Add a new plug-in to an existing UAB component integrating it in the component's wizard.

These new features will allow developers to focus on implementing the desired functionality for the UAB component by extending an existing structure and, therefore, reducing the amount of time required to develop new UAB components.

CONCLUSION

The use of continuous integration tools have been proven to be a best practice in software engineering during the software development and maintenance phases and assist with higher software quality [6].

The UAB development team has made an effort to apply the same practices to the development of control system applications, providing tools to perform automatic executions of the code generators or plug-ins. In the scope of UNICOS applications development, these tools allow to reduce the user interaction and avoid repetitive, error-prone and time consuming tasks. In the case of UAB software development, the tools provide a mechanism to verify the correctness of the modifications introduced in the software by comparing the generated outputs with previous versions.

These tools are currently being used when developing UNICOS-based control systems. The LHC GCS (*Large Hadron Collider, Gas Control System*), LHC Cryogenics and many HVAC (*Heating, Ventilation and Air Conditioning*) control applications are some examples of them. In summary, there are more than 50 applications using these tools which generate more than 14 million lines of PLC code.

REFERENCES

[1] Ph. Gayet, R. Barillère. "UNICOS a framework to build industry like control systems: Principles & Methodology".*10th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Genève (Switzerland), 10-14 Oct 2005.

[2] B. Fernández Adiego, E. Blanco Viñuela, I. Prieto Barreiro, "UNICOS CPC6: Automated code generation for process control applications", *13th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systems.* Grenoble (France), 10-14 Oct 2011.

[3] M. Dutour. "Software Factory Techniques applied to process control at CERN". *11th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systems.* Tennessee (USA), 15-19 Oct 2007.

[4] Hervé Milcent, Enrique Blanco, Frédric Bernard, Philippe Gayet. "UNICOS: An open framework". *12th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systems.* Grenoble (France), 12-16 Oct 2009.

[5] Apache Maven Project, `http://maven.apache.org/index.html`

[6] Martin Fowler, "Continuous Integration" (2006), `http://martinfowler.com/articles/continuousIntegration.html`

[7] JUnit Framework, `http://junit.org`

ISBN 978-3-95450-139-7