Generating Safety-Critical PLC Code From a High-Level Application Software Specification

Seamless Command and Control Coordination

The benefits of automatic-application code generation are widely accepted within the software engineering community. These benefits include raised abstraction level of application programming, shorter product development time, lower maintenance costs, and increased code quality and consistency. Surprisingly, code generation concepts have not yet found wide acceptance and use in the field of programmable logic controller (PLC) software development. Software engineers at Kennedy Space Center recognized the need for PLC code generation while developing the new ground checkout and launch processing system, called the Launch Control System (LCS). Engineers developed a process and a prototype software tool that automatically translates a high-level representation or specification of application software into ladder logic that executes on a PLC.

All the computer hardware in the LCS is planned to be commercial off the shelf (COTS), including industrial controllers or PLCs that are connected to the sensors and end items out in the field. Most of the software in LCS is also planned to be COTS, with only small adapter software modules that must be developed in order to interface between the various COTS software products.

A domain-specific language (DSL) is a programming language designed to perform tasks and to solve problems in a particular domain, such as ground processing of launch vehicles. The LCS engineers created a DSL for developing test sequences of ground checkout and launch operations of future launch vehicle and spacecraft elements, and they are developing a tabular specification format that uses the DSL keywords and functions familiar to the ground and flight system users. The tabular specification format, or tabular spec, allows most ground and flight system users to document how the application software is intended to function and requires little or no software programming knowledge or experience. A small sample from a prototype tabular spec application is shown in Figure 1.

LINE	ROUTINE	DSL API	OBJECT(S)	DESCRIPTION/MESSAGE	LO/VAL	HI	VOTING	DURATION	REACTION
1	*# Routine sends primary OPEN command and waits for appropriate indicators*								
2	**OpenValve1Primary**	*# Send primary OPEN command*							
3		**send_command**	VLV1_PRI_OPEN_CMD	Valve1 Primary Open Command	ON				
4									
5		*# Verify both primary indicators change appropriately within appropriate time durations, on failure call another routine to perform Secondary Open Command*							
6		**verify_within**	VLV1_PRI_CLOSED_IND	Valve1 Primary Closed Indicator	OFF		2 of 2	8 sec	
7		**verify_within**	VLV1_PRI_OPEN_IND	Valve1 Primary Open Indicator	ON		2 of 2	26 sec	**OpenValve1Secondary**
8	end								
9									

Figure 1. Sample of tabular spec formatted application.

The LCS developers needed a mechanism or tool to translate application software from tabular spec format into PLC code to execute on the PLC platforms out in the field. The functionality of some representative samples of tabular spec was manually coded into PLC ladder logic and tested with a field item simulator to verify the proper operation of the manually coded ladder logic. This manual process of conversion or translation from tabular spec representation to PLC ladder logic demonstrated that translation points or patterns existed between portions of the tabular spec and portions of the PLC ladder logic.

With the aid of these translation points, a few representative samples of the manually coded PLC ladder logic were exported from the PLC coding Integrated Development Environment (IDE) as plain text. This exported text was then converted by hand into plain-text PLC code "libraries" with the intent that a future automatic utility to translate tabular spec to ladder logic would use these PLC code libraries. After some representative samples were translated from tabular spec to PLC ladder

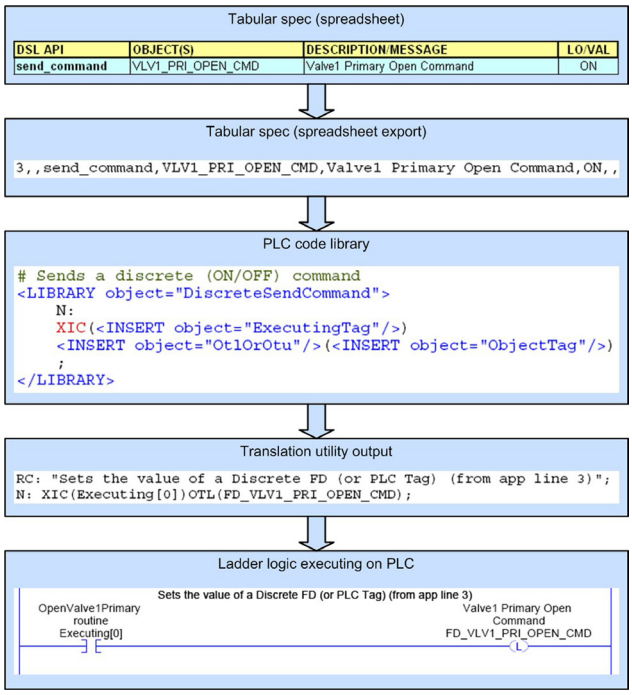

Figure 2. Transition process flow.

logic and after a PLC code library was manually created for each of the translation points contained in the samples, a prototype automatic translation utility was developed and demonstrated. The translation capabilities of the utility will be expanded upon as more and more translation points are identified, translated manually, and then turned into additional PLC code libraries.

Figure 2 shows the simplified process flow for translating a single "send_command" line from tabular spec to PLC code. The spreadsheet that contains the application software in tabular spec format is exported to plain text and is used as input to the translation utility, along with the PLC code library. The translation utility processes these input files, using program transformation steps. This creates an output file that can be imported by the PLC coding IDE.

This work successfully demonstrated that a process and a software tool can generate executable PLC code from a high-level specification representation of a safety-critical control system. This process includes some manual work to find translation points and to create PLC code libraries, but that up-front and one-time manual effort is overshadowed in the end by the automatic generation of repetitious and tedious functionality that would be difficult and error-prone to perform manually. Such a process and tool increase the quality, reliability, maintainability, and verification/validation pedigree of the PLC code over code that is generated manually. It also provides a high level of PLC code consistency and could even reduce operations and maintenance costs for the control system after it is deployed.

Follow-on phases of development of the automatic translation utility should include most, if not all, of the following tasks:

- represent as much LCS application software in the tabular spec format as possible without overcomplicating the tabular spec format,
- manually implement the remaining translation points and any newly discovered translation points, along with the matching PLC code libraries,
- add code to the translation utility to recognize and handle the new translation points, along with the new PLC code libraries, and
- test and certify the translation utility for automatic generation of safety-critical PLC control logic in the LCS at KSC.

Contacts: Kurt W. Leucht <Kurt.W.Leucht@nasa.gov>, NASA-KSC, (321) 861-7594; and Glenn S. Semmel <Glenn.S.Semmel@nasa.gov>, NASA-KSC, (321) 861-2267