A Software Approach for the Design of a Virtual Plant Generator

Lucien Ngalamou† and Leary Myers††

†School of Engineering, Grand Valley State University, 301 W. Fulton, Grand Rapids, MI 49504
††University of the West Indies, Mona Campus, Kingston 7, Jamaica

Summary
This paper presents the design considerations of a Virtual Plant Generator (VPG), which seeks to address the need for a simplified, yet effective method to represent the virtual models of control/automation processes, thus providing a mean by which the control logic of a system can be visually emulated. This method eliminates the need for testing control logic on actual plant equipments, by replacing them with a virtual model constructed using VPG. VPG is a component of a software tool called Industrial Process Control Studio (IPCS) dedicated to industrial process control and automation. IPCS is organized in four major modules according to the criteria of modeling, supervision, human-machine interfacing, and virtual representation

Key words:
Virtual plant generator Virtual model, Emulation, Industrial Process Control Studio

1. Introduction

In a broad sense a programmable logic controller (PLC) can be defined as a microcomputer as well as a microcontroller [1], [2]. They both comprise the same basic components: an arithmetic and logic unit (ALU), a control section, a local memory area, and input/ output (I/O) ports. They are however usually less powerful than a personal computer. Some common PLC applications are:

- batch processing and material handling in the chemical industry,
- machining and test stand control,
- data acquisition in the manufacturing industry,
- wood and chip handling in the lumber industry,
- filling and packaging in the food industry, and
- furnace and rolling mill controls in the metal industry [1], [3].

PLCs are now used to replace traditional relay-based controllers that were commonly found in control industrial processes as they offer more flexibility. The menu of options that contribute to this flexibility include: the small size of the PLC facilitates locality of controller with the machine or the process being controlled, user friendly computer software to allow specification of the control process in the standard methods of programming [3] whether Structured Text (ST), Function Block Diagram (FBD), Instruction List (IL), Sequential Function Chart (SFC), Ladder Diagram (LD) or other modern programming constructs such as state diagrams, networking in local area networks (LAN) which allows for remote management of control processes through communication ports and the standardization of hardware interfaces for manufacturer interchangeability [1], [3], [4]. To effectively deploy PLCs in any process control or automation endeavor, there is a requirement for the integration of elements such as electronic circuits, sensors, actuators, and software integrated development environments (IDEs). An IDE generally consists of a graphical user interface (GUI), a machine code generator/interpreter, a simulator/debugger, and a loader. The design of IDE tools for PLCs requires a good knowledge of visual programming, software engineering, and system programming [4], [5]. It might be interesting to consider that a PLC or any other device to be used in a control process should be seen as a resource and it is when the control problem is clearly modeled that a choice of PLC is made. Reasoning alone this line, a research project was initiated that consists of analyzing and developing an industrial process control tool for efficient deployment of PLCs in single or distributed modes [23]. By deployment we mean all the steps that consist of control specification, model capture of the process using IEC programming languages, formal verification, resource allocation, virtual plant modeling, supervisory control, and code generation. Process Automation may be conceptualized in terms of above functional modules. Each of these modules is responsible for the execution of some particular program functionality, and collectively, they function as the Industrial Control Studio. Figure 1 below is a block diagram of the Industrial Process Control Studio, and actually serves as a template for the actual implementation of the control studio.

It is to be noted that all the modules provided for the control studio consist of the following:

- Visual Plant Generator
- Formal Verification
- Schematic Generator
- Resource Allocation and Database
- Simulator and Real-time Debugger
- Report Generator

Manuscript received September 5, 2010
Manuscript revised September 20, 2010

- Retargetable Code Generator
- IEC 61131-3 Model Capture
- Scada Module1,
- Human-machine Interface (HMI)
- XML Engine and Synchronizer.

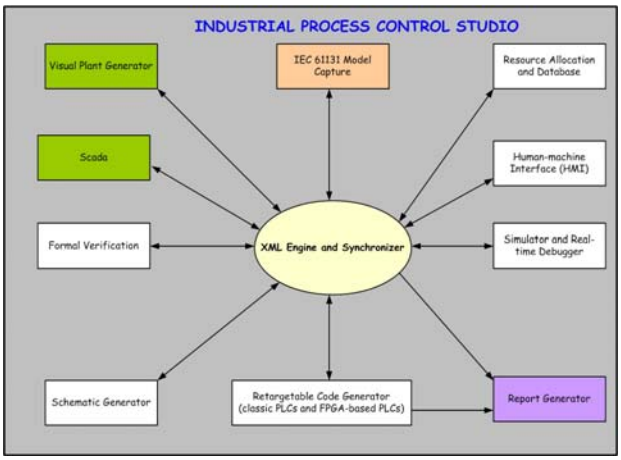

Fig. 1. Block Diagram of the Industrial Process Control Studio (IPCS) Tool

These modules are further organized in four major groups according to the criteria of modeling, supervision, human machine interfacing, and virtual representation.
The complete deployment of ICPS in a distributed control environment uses concepts similar to the Lintoutch [12] (Figure 2), which is an open source HMI software for real-time monitoring of process automation. The development of the following elements are considered:

- **IPCS IDE (Integrated Development Environment)** is used to capture the model of control problems and the deployment PLCs. Each control model may have HMI, Scada, VPG, and Simulation/Debugging Modules as part of a project which can later be deployed to the IPCS Server.
- **IPCS Runtime** downloads the deployed ICPS Project, visualizes the screens, and sends user generated changes to the IPCS Server. The communication protocol used to exchange data among IPCS Server and one or more IPCS Runtimes is a custom made and documented protocol built on top of TCP/IP.
- **IPCS Server** connects to the monitored systems via special industrial protocols and sends changes of the data to the IPCS Runtime. New communication protocols be plugged to the IPCS Server by developing new Server Plugins.

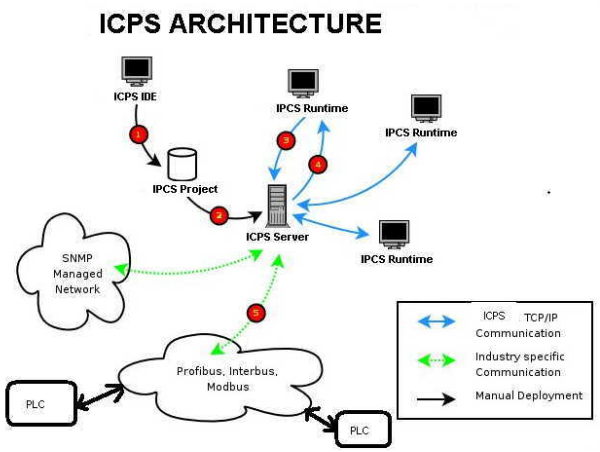

Fig. 2. IPCS Architecture

This paper analyses the design of the virtual plant generator (VPG) module of the IPCS. The design and implementation of VPG seeks to address the need for a simple yet effective method of modeling control/automation systems. In addition, the software must also be time saving and cost effective to allow fast simulation of PLC control logic code. These goals are intended to be accomplished through the elimination of testing control logic on actual industrial plant equipment. Instead, the PLC control logic will be tested within a virtual environment which is expected to give a general understanding as to the validity of the control logic code.

A review of existing tools leads us to two main software:

- General Electric's DataViews [5]
- Festo Didactic's EasyVeep [6]

Although the two software systems do not possess exactly what the VPG is intended to do, they have similar attributes.

DataViews is a graphics solution for programmers interested in developing their own Human Machine Interface. It consists of customizable graphic development tools that allow the creation of robust, dynamic graphics within any application design. The integration of these graphics within the application is done at a much shorter time than it would take a programmer to program them individually. Such time conservation allows programmers to focus more on the overall application design. The design of the Virtual Plant Generator is intended to eliminate the need for having to design graphics but to have the graphics readily available for control system designs.

This eliminates the need for the use of extensive coding of the system graphics design. However, the design only permits the specifications of the control models that are

designed for the Generator. Such a design limits the flexibility of the graphics but the intention is to build the Generator to model abstract control systems so that there is no need for the modeling of the objects to vary beyond what they are programmed to do. As also outlined, the Virtual Plant Generator is intended to allow for rapid prototyping on programmable logic controllers hence the elimination of graphics designs via coding would decrease the amount of time spent on graphical model designs and increase the time spent on performing other tasks relevant to control system designs.

EasyVeep (Easy Visualized Equipment Emulation Program) is a software package specifically designed to aid in the programming of PLCs. It perhaps possesses the closest similarity to the Virtual Plant Generator. It is specifically designed to allow for control logic to model defined control/automation systems. There are twenty eight defined virtual control systems designed within it. Operation of the software is conducted by utilizing external hardware devices and control logic language tools. The simulator allows for the PLC to control the virtual environment within EasyVeep rather than a real control system as it is designed for as supported by VPG. EasyVeep can also be utilized to download more virtual designs that have been conceived by FESTO [8]. Online requests may also be made for ideas on more virtual control system designs.

The rest of this paper is organized in four sections. Section 2 describes The design concept and implementation of the Virtual Plan Generator followed by section 3, which presents the testing. Section 4 deals with the conclusion and future work.

2. Design Concepts and Implementation of VPG

2.1. Design Specification

The Virtual Plant Generator is intended to allow the design of any abstract virtual control system. After the system design, control logic code is expected to be conceived and input to VPG, which then execute on the designed virtual system. The operation of the code can then be verified based on the system response. Virtual control systems are not expected to function in unison when constructed. Instead, the system is to be strictly based on control logic being used to control it. Individual models are however, expected to function on their own when simulated. Actuators will not be expected to possess the ability of being controlled automatically. Instead, they should be manually controlled to indicate responses within the system. This gives the programmer the opportunity to interact with the system and so be able to determine whether or not the system responds as is expected. The designs of control/automation scenarios will be limited by the models and the programmer's imagination. Programmers utilizing VPG will experience time saving during the testing of control logic since the VPG will not require any hardware interfaces to function. Instead, VPG will function as a virtual PLC and environment, being able to interpret control logic and simulate it within the virtual plant designs.

The following features of the VPG are expected:

- Time Efficiency - Efficiency is sought through the elimination of code within graphical model designs and the elimination of external hardware connections for operation.
- Cost Effectiveness - Reduced cost is expected as a result that no hardware required for operation.
- Ease of use - Easy usage is considering during the design of the GUI which is intended to be user friendly.
- Increased Productivity - With good time efficiency, more time can be available for design of control logic or more control/automation system designs.

2.2. Design Considerations

2.2.1. Choosing the Software Tool

In deciding how to implement a prototype of the Virtual Plant Generator, the choice of the most appropriate software tool was made. This was done by considering two software development tools: Macromedia Flash [18] and Qt [19]. An overview of the research done into the Macromedia Flash and QT will now be given to show the viability of using these development tools for the modeling of the virtual objects.

2.2.2. Considering Macromedia Flash

Macromedia Flash [19] is a development tool that is geared mainly towards creating interactive web site graphics. "Flash was originally just a vector animation tool but is now one of the most advanced programs for creating rich Internet applications to provide powerful user experiences [14]". The suitability of Flash for web graphics lies in its ability to create animations with small file sizes. These small file sizes are attributable to the fact that it uses vector graphics. Vector graphics can be scaled to any size without causing pixel alteration, a flaw that is associated with the resizing of ordinary bitmap graphics. Vector graphics are also very small in size, because they are generated mathematically. As an example,

to draw a circle, Flash will only store the circle's center point and radius. Using this information it will compute which pixels to draw in order to produce the circle. On the other hand, bitmap (raster) graphics stores information about every point (pixel) in an image. As a result the file size for bitmap graphics is generally much larger than that of vector graphics, and bitmaps tend to look jagged or pixelated when scaled. Bitmaps are more appropriate for photographic images that can't be described easily with vectors [15]. For the implementation of interactivity in the application the use of a language known as Action Script is necessary. "Action Script is the language you use to add interactivity to Flash applications. You don't have to use Action Script to use Flash, but if you want to provide basic or complex user interactivity, work with objects other than those built into Flash (such as buttons and movie clips), or otherwise turn a SWF file into a more robust user experience, you'll probably want to use Action Script [16]". In order for the modeling of the various objects of an automation process to be achieved, with the use of Macromedia Flash, the need for the use of the Action Script language is imperative. Using Macromedia Flash for the design VPG requires the use of two programming languages will have to be utilized; one for the creation of the GUI (Graphical User Interface) which will provide the interfacing of the application with external parameters (user's files, signals), and the other language for the modeling of the Virtual Plant objects.

2.2.3. Considering Qt

Qt [18] is a C++ framework for developing cross-platform GUI applications. Qt allows programmers create applications that will run on Windows, Mac OS X, Linux, Solaris, HP-UX, and many other versions of Unix with X11. The use of Qt on a single platform in no way undermines its performance as the numerous features that this development framework provides are quite capable of competing with the more popular GUI development applications. Among the uses of Qt include the creation of sophisticated software systems, such as 3D animation tools, digital film processing, electronic design automation (for chip design), oil and gas exploration, financial services, and medical imaging [17]. Among the benefits that Qt provides are its single-source compatibility, its feature richness, its C++ performance, the availability of the source code, its documentation, and the high-quality technical support. For the rendering of images, as needed for the virtual representation of objects, Qt provides a very powerful 2D Paint Engine that not only allows the creation of various shapes and complex paths, but also allows the drawing of popular picture files such as JPEG and PNG files. With respect to the adding of interactivity to the objects, the C++ style classes provided are more than capable of creating structures that would control the properties of the objects. As far as the animating of the objects are concerned, the use of timer events has been suggested in Qt literature as a viable solution for producing animated displays.

2.2.4. Macromedia Flash versus Qt

In considering the merits of the two software tools contrasted, it was ultimately decided that the use of the Qt software development application was the most appropriate choice for the modeling of the virtual objects and its system. This was primarily due to the fact that the graphic intensity required by the Virtual Plant Generator can be produced using both of the above named tools, however, the simplification that Qt provides by integration into the target development language gives incentive for its use in this project. This underscores the fact that the need to learn additional languages will not be necessary as Qt is a C++ based framework, and all the necessary programming that needs to be done in order to complete the Virtual Plant Generator can be conducted from one programming environment.

2.2.5. Design Approach

The concept of animation is quite common to the design of any animated film. It is based on sequentially displays two-dimensional graphics with the effect of creating motion. Such motion is popularly displayed as video files. For the purpose of the VPG, the design of animated graphics is done utilizing this animation method. The general model picture is first designed within the Adobe ImageReady Tool[?]. From here, the picture is placed onto an empty picture window where Qt functions are utilized to paint onto it. The painting occurs sequentially to form the model picture and hence creates various frames which are stored within a list of pictures. This list is sequentially walked through to create animation.

2.2.6. Data Coding for VPG

An extended and more elaborate use of the Virtual Plant Generator is to represent the virtual model of a control/automation process being supervised by a PLC or a group of PLCs. In this regard the control logic of the system is input directly into the application to control the animated response of the Virtual Plant, instead of the use of a control file.

There are two approaches to designing the Virtual Plant Generator to operate on PLC control logic:

1) Designing the software to interpret PLC output signals

2) Designing the software to interpret the logic code input to the PLC

In making the decision as to which approach to take to the data coding for VPG, taking into account the need for a cost effective and time saving method, it was concluded that the second choice was the most appropriate. Interpreting control logic input to the PLC allows the development of control logic without the need for a PLC to test the code. Basically, the designed software is intended to be a virtual PLC allowing itself to be programmed and used to control an abstract virtual control/automation system. This is considered more conservative as compared to utilizing a PLC. PLCs are costly and hardware based, allowing for less effectiveness in simulating control logic quickly. Reading PLC control logic, on the other hand, simply requires a method of being able to interpret what the logic means without any hardware interaction. This would allow users to test control logic at anywhere and anytime without the need for a PLC to interface with. All that would be needed is a Control Logic Language Development Tool and the Virtual Plant Generator. This now leads to another important task which requires the evaluation and selection of an appropriate PLC control logic development tool. To select the development tool, an analysis was first needed to determine the appropriate PLC language that the Virtual Plant Generator would be simulating. The IEC 61131-3 [2] standard details the different forms of control languages that are utilized and hence allows for the determining of the appropriate language to be utilized. There are five different programming language standards. Two are textual whilst the remaining three are graphical: Textual Languages (Structured Text and Instruction List (Statement List)); Graphical Languages: (Function Block Diagram, Ladder Logic Diagram, and Sequential Function Chart). Logically, it would be considered easier to interpret textual languages rather than design software to interpret graphical code. How- ever, the most popularly utilized method of PLC programming is the Ladder Logic Diagram. The reasoning goes back to before the development of PLCs where control systems were designed using relays. The relays however, could only be used for making simple logic decisions in control systems. As the PLC came into the spotlight in the 1970s, engineers now had to cope with programming them as compared to designing and implementing relays to control systems. To reduce the need to retrain engineers and trades people, the ladder logic language was developed which purposely mimics relay logic within its language. This eventually made ladder logic the most utilized control logic language [6]. Despite this language being the most popular, however, it still makes the development of the VPG software system difficult compared to utilizing textual languages. Nonetheless, ladder logic was considered the primary language that needed to be interpreted by the VPG. Following this decision, there was the need to determine the type of PLC that the software should be designed for. There exists many PLCs, however, there are some that are more widely utilized as compared to others. This was considered the most appropriate approach to selecting a type of PLC. Siemens, Allen-Bradley, ABB, Mitsubishi, Omron, and General Electric are all popular brands of PLCs but to reduce the scope, a decision needed to be made. This decision consisted of determining which type of PLC language development tool was capable of programming in multiple control logic languages and capable of converting ladder logic language into a type of textual language. Allen-Bradley's "PLC Communicator DEMO" can be utilized to create ladder logic diagrams but is unable to convert the language to an appropriate textual form. Siemens SIMATIC "STEP 7 MicroWIN" [22] is capable of creating ladder logic diagrams and also has the ability to convert this graphical code into a textual one known as an Instruction List language or Statement List. In addition, familiarity has already been established with this software and is considered quite easy to utilize. The PLCs programmed by this software tool are quite widely utilized in industry. One such example is the S7-200 which is a micro PLC. It has a small size, is designed similarly to a brick and consists of a power supply and input/output modules on-board. In addition, it has the simplicity of operating on stand alone applications and may yet even be used to control/automate even more elaborate and complex control system designs. Having chosen the software to develop the control logic with the most effective methodology, there was the need to analyze the instruction set for the IL language. Considering the time allotted for the development of the VPG, it was necessary to determine the most important instructions to design the system to interpret. These are basic logic statements allowing models to be controlled by the status of other models.

2.3. VPG Architecture

The design of VPG requires the use case UML diagram that gives a representation of what the system is expected to do. Here, the actor is able to interact with the VPG system via the activities of the system. The system's activities offer more flexibility and shows the levels at which the actor is allowed to interact with the system. In return, these system activities allow for screen output changes as well as the ability to create and modify file formats specific to the operation of the VPG. At any one point in time, there is only expected to be one actor

(client) interacting with the system thereby allowing the system to be a stand alone software tool capable of being operated by a user who is competent in the application's functionality.

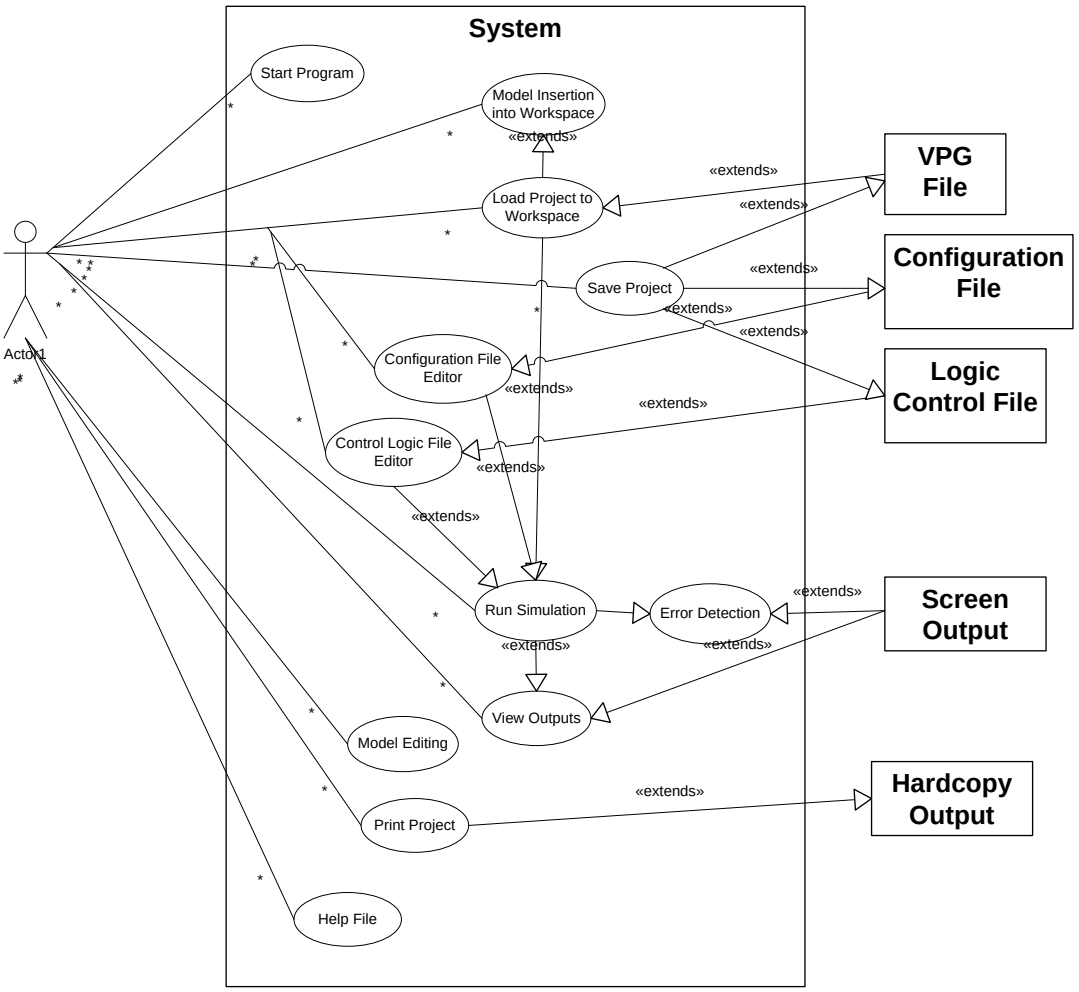

Fig. 3. Use Case Diagram of the VPG

A typical narrative operation of the system illustrated in Figure 3 is described as follows:

1) The user starts the software GUI.
2) The user selects the workspace tabulation window.
3) The user selects the control model option in the toolbox window by clicking it once.
4) The user inserts components/models into the workspace by clicking the location for the model to be placed.
5) By right-clicking the models within the workspace, they can be edited to suit the user.
6) The control logic tabulation window is selected.
7) Control logic STL is created within the Control Logic File Editor or pasted from a control logic development tool.
8) The configuration tabulation window is selected and the names of the workspace models are assigned the variable names within the STL code.
9) The project is saved.
10) Simulation is begun by selecting "Run Simulation".
11) In the event of errors within the execution of the code onto the workspace environment, prompts are made to indicate the need for remedial measures.
12) Steps are taken to troubleshoot the code and the virtual environment.
13) With no errors, the simulation commences allowing the user to activate discrete inputs and view discrete outputs.

The potential classes for the application were brought about by analyzing the conceived design

of the VPG. Table 1 provide the initially deduced attributes of the software classes.

Table 1: Potential Classes and attributes of the VPG

Class	Attributes
VPG Main Window	VPG Model Toolbox, Help Window, Tab Window
VPG Model	Motor Model, Tank Model, etc.
VPG Model Toolbox	Motor Model, Tank Model, etc.
Workspace Window	VPG Models, LCF Control Model, VPG File, Control File, Configuration File
Control Logic Window	Control File
Configuration Window	Configuration File
Help Window	Documentation
Tab Window	Workspace Window, Control Logic Window, Configuration Window
LCF Control Model	Dependents, independents

The diagram of Figure 4 shows the individual classes and how they relate to each other. Each class consists of attributes of which operations are performed on so as to produce new data. The data is then saved as attributes of a particular class. Each operation is part of a class and so entities can be divided into classes, operations and attributes. The overall diagram highlights the work previously done in addition to the ideas conceptualized to improve the software. Almost all entities have been modified and as such, there is need for the documentation of all changes made to the prototype model of the VPG.

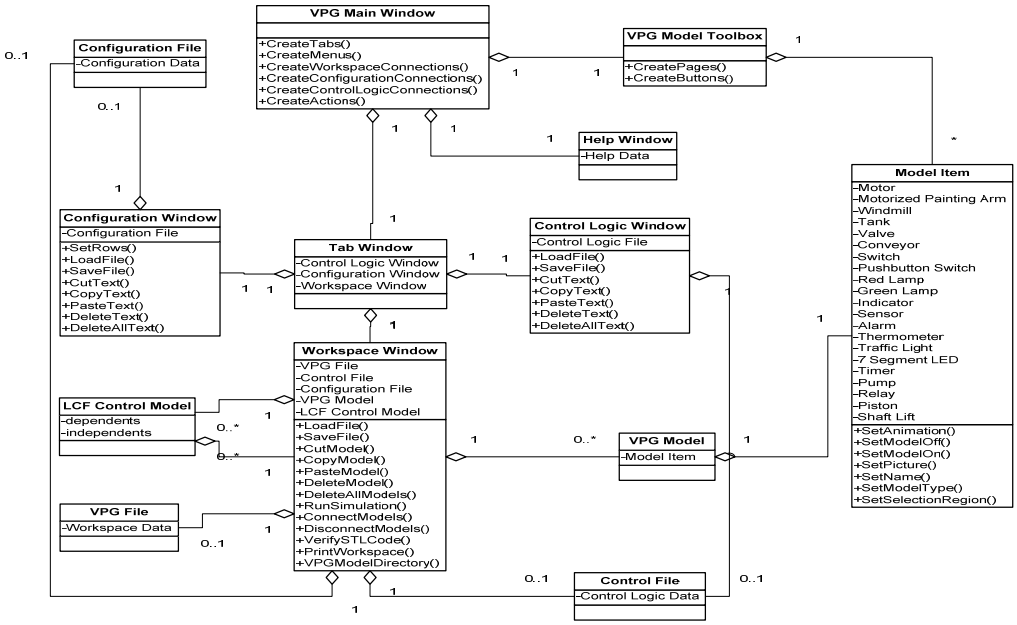

Fig. 4. Static UML Class Diagram of the conceived classes

These concepts were implemented as the Virtual Plant Generator software tool. An overview of its user interface is given in Figure 5. With respect to the workspace environment, all conceived ideas were unable to be shown within this design approach but they are documented within the program design and the implementation of the workspace class.

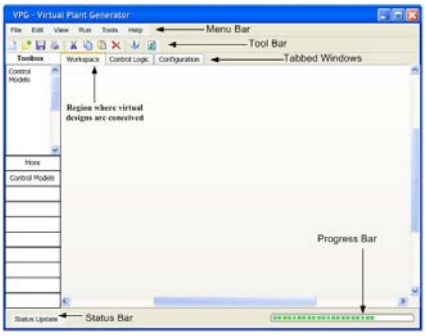

Fig.5. An Overview of the VPG's GUI

The control logic tabulation gives the user access to an

environment capable of designing control logic code based on the instruction set designed for. This environment is not expected to be where the code will be created. Instead it is expected that a control logic language development tool will be used to create the logic and then the logic be copied within this window. For users familiar with the language however, the code can be typed here provided it is within the limitations of the instruction set. The configuration window allows for the control logic variables to be assigned to the models within the workspace. The technical jargon for the design of control logic is usually Qx.x for outputs and Ix.x for inputs as stated within the ”Communication Protocols” section of [20]. This window basically provides an interface for the workspace window and control logic window; allowing users to utilize code created by programmers and implement it in the designed virtual environment.

3. Testing and Evaluation

For the of any new technology, be it hardware or software; it is a standard procedure to apply testing and evaluations. Super classes were developed first, it was possible to apply a defensive programming techniques for other classes. This approach accounts for the immediate testing of any function following its implementation. As such, the functions could be evaluated to ensure that they do indeed perform as specified. The most important concerns the memory leakage during operations. Table 2 shows the non existence of such phenomena during VPG operations

Scenario 1: The VPG prototype is started and basic operations done whilst monitoring the memory usage of the system.		
Input	**Result**	**Reasoning**
Software application is started.	Memory usage stands at approximately 65 Mb	Memory usage is expected to remain fairly constant since the animation frames for each type of model is created when the system initializes.
Several models are inserted into the workspace, STL code is added and configurations made	Memory usage stands at approximately 65 Mb	Since the insertion of models is the main reason for a lot of memory usage, allowing one list of animations for each type of model allowed for the memory to not vary at all.
A new project is loaded	Memory usage increases to approximately 70Mb (See Figure 26 of Appendix D)	The memory usage was expected to remain constant but it increased by 5Mb. This was however considered small and insignificant since no more change in memory was observed. Research is however, expected to be done on this issue at a later date.
More projects are opened and models added to them	Memory usage stands at approximately 70 Mb	The 5 Mb seems to only occur on the first opening of a project but afterwards, the memory holds constant at 70 Mb.

Table 2: The VPG prototype is started and basic operations done whilst monitoring the memory usage of the system

Other tests, which are documented in [20], were performed to prove the reliability and the stability of the VPG software tool. Figure 10 shows one of the screen shots relating a step in the design of the virtual model of a thank system.

4. Conclusion and Future Work

Following the final testing of the Virtual Plant Generator, its design was considered a viable venture. Research has thus far indicated the lack of any software tool similar to VPG. In order to justify the flexibility of the software, several control/automation were designed and programmed for simulation using Statement List (STL) instructions. The results obtained further justify the development of such software tool.

VPG as part of IPCS possesses the potential to be quite an attractive software tool for PLC technology. There are however, several useful improvements that can still be made to make it a success story, they are listed below:

1) Control model animations based on the system;
2) Ability to interpret the entire ser of the instruction list;
3) More models to be added to the software;
4) Ability to interpret more than one control language.

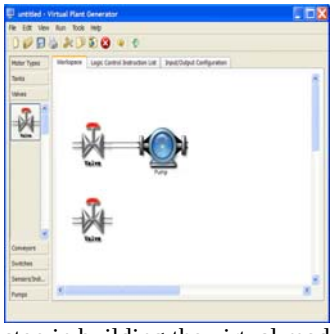

Fig. 7. A step in building the virtual model of a tank system

References

[1] I. Warnock, Programmable Logic Controllers - Operation and application, Prentice Hall, Englewood Cliffs, NJ; 1998.
[2] ICE (International Electrotechnical Commission), IEC Standard 61131-3: Programmable Controllers - Part 3, 1993
[3] T. Mandel The Elements of User Interface Design, John Wiley & Sons, 1997.
[4] Marriot, K. and Myers B. (1998), 'Visual Language Theory', Springer.
[5] General Electric Industrial Systems (GE) (2010), 'www.gefanuc.com', Accessed September 25, 2010.
[6] EasyVeep, http://www.easyveep.com, accessed September 25, 2010.
[7] Beck, L. L. (1997), 'System Software: An Introduction to Systems Programming' Addison Wesley.
[8] Festo Didactic (2010), 'www.festo-didactic.com', Accessed September 25, 2010.
[9] Vogeler, D., Wilson, E., and Barber, L. (2005), 'Macromedia Flash Professional 8 UNLEASHED', Sams.
[10] Lott, J. (2006), 'Chapter 1. Drawing in Flash in Flash 8 Cookbook"', O'Reilly pp. 1-36.
[11] Dehaan, J. and Dehaan, P. (2005), 'Introduction in Learning Action- Script 2.0 for Macromedia Flash', Macromedia, pp. 1-6.
[12] Schmitt-Walter Automation Consult (SWAC) (2010), 'http://lintouch.org', Accessed September 25, 2010.
[13] D. Boley and R. Maier, "A Parallel QR Algorithm for the Non-Symmetric Eigen value Algorithm", in Third SIAM Conference on Applied Linear Algebra, Madison, WI, 1988, pp. A20.
[14] D. Vogeller et al, "Macromedia Flash Professional 8 UNLEASHED", Sams, 2005.
[15] J. Lott, "Chapter 1. Drawing in Flash" in Flash 8 Cookbook, O'Reilly, 2006, pp. 1-36.
[16] J. Dehaan and P. Dehaan, "Introduction" in Learning ActionScript 2.0 for Macromedia Flash, Macromedia, 2005, pp. 1-6.
[17] J. Blanchette and M. Summerfield, "Chapter 8. 2D and 3D Graphics" in C++ GUI Programming with Qt 4, Prentice Hall, 2006, pp. 47-52.
[18] Qt, http://www.trolltech.com
[19] Macromedia Flash, http://www.adobe.com/products/flash/
[20] V. Sahatoo, "Virtual Plant Generator, Final Report for ECNG 3020: Special Project for BSc. Electrical Engineering", Main Library, Uni- versity of the West Indies, St. Augustine, Trinidad and Tobago.
[21] Adobe ImageReady Tool, http://www.adobe.com, Accessed September 25, 2010.
[22] Siemens Energy & Automation Inc. (2010), www.sea.siemens.com/step/pdfs/plcs.pdf, Accessed September 25, 2010
[23] L. Ngalamou and Leary Myers, "Modelling PLC Characteristics for Resource Allocation", Int. J. Computer Applications in Technology, Vol.31, No 3-4, 2008 , pp.263 - 274.

Lucien Ngalamou received the B.Sc.(First Class Honor) Degree in Applied Physics from the University of Yaounde, Cameroon in 1989 and the Master Degree in Electronic Engineering from the University of Science and Technologies of Languedoc, Montepellier-France in 1991. He completed his PhD in Electronic Engineering from Joseph-Fourier University, Grenoble-France. He is presently an Assistant Professor in the School of Engineering, Grand Valley State University, Grand Rapids - Michigan. His current research interests include Reconfigurable Computing, Electronic Systems Design, CAD Tool for Process Automation, Formal Hardware Verification, Evolvable Hardware, Asynchronous Logic, and Models of Computation.

Leary Myers is a Lecturer in the Department of Physics, University of the West Indies – Jamaica and holds a PhD in Electrical Engineering from the Howard University, Washington DC. He has occupied such positions as Director - Five Star Engineering and Scientific Associates Ltd. He was a Part-time (Senior) Lecturer, School of Engineering, University of Technology - Jamaica; Research Assistant Professor, Graduate School of Arts and Sciences; and Senior Research Associate, Materials Science Research Center of Excellence at Howard University.