SAND2016-2041C

Dynamic Radiography Acquisition Software System for Computed Tomography Applications

Rahul Nunna[1], Ismael Perez[1], Kyle R. Thompson[2], and Edward S. Jimenez[1]

[1]Sandia National Laboratories, Software Systems R&D
PO BOX 5800, Albuquerque, NM 87185-0933, USA
(505) 284-9690; email esjimen@sandia.gov

[2]Sandia National Laboratories, Structural Dynamics and X-Ray/NDE
PO BOX 5800, Albuquerque, NM 87185-0555, USA
(505) 844-0347; email krthomp@sandia.gov

ABSTRACT

This work aims to deliver a hardware-specific image acquisition software environment for use in direct radiography and computed tomography applications. The goal of this project was to develop a dynamic control application that provides the necessary functionality relevant to industrial computed tomography applications, as well as full compatibility with many manufacturers of radiography and computed tomography system hardware currently available on the market. Several object oriented design patterns were adopted in order to develop a system capable of communicating not only with different manufacturers of hardware, but also communicating with various species of hardware (e.g. motor controllers, X-Ray detectors, X-Ray sources). The system also enabled a channel of communication between each of the various hardware systems. This was done by generalizing each of the hardware manufacturers' control libraries to a set of generic hardware control libraries. Because of this, new hardware control libraries can easily be imported into the system, with little to no adjustment in source code. The proposed software system will decrease the need for long term software maintenance, thereby lowering development costs. This system will in turn allow for the introduction of new innovative configurations of computed tomography scanning systems, which may open new possibilities for radiography research.

INTRODUCTION

An industrial x-ray imaging system generally consists of an x-ray source, x-ray detector, and an object of interest which is mounted to a multi-motor staging system. Each subsystem of the x-ray imaging system can be connected to an arbitrary number of stepper motors controllers. To add to the complexity, each of the motor controllers may potentially be from different manufacturers. Each hardware manufacturer has its own specific software libraries which can be used by developers to send commands to the motor controller in order to perform a specific task. One possible configuration is that each part of the system (i.e X-Ray source, X-Ray detector, and mounting stage) has motor controllers made from different manufacturers. In order to address this, the Dynamic Radiography Acquisition Software System (DRASS) will be able to easily incorporate software libraries from different manufacturers into one cohesive software application. To our knowledge there is no software in existence which can accommodate the needs to have motor controllers from different manufacturers for complex configurations. Some works have developed systems that are application specific. In the work of Wang S. et al, for example, their x-ray imaging system incorporated an X-Ray phase contrasting system. This system uses 22 motor controllers, from the same manufacturer [1]. Although the number of motor controllers is not typical for traditional X-Ray systems, the functionality to be able to accommodate motor controllers from different manufacturers is not possible without making significant adjustments to their software.

Ortega G.A. developed a software system for a micro-CT system; the software developed for the thesis is similar to DRASS, but their work tries to overcome the issue by attempting to generalize the software for incoming hardware that they were expecting to get in the future, which is not necessarily just for motor controllers [2]. Technosoft is the only motor controller manufacturer that is mentioned in her work, which might not be able to accommodate for other motor controller manufacturers. Although this work only uses one motor controller manufacturer, it uses a similar approach by developing classes for each of the hardware controllers' respective APIs. With DRASS, it is possible to construct any configuration of motor controllers from different manufacturers for an X-Ray imaging or CT system. The next section will provide further details of the ~~proposed~~ software system.

METHODS

The software was developed as per the following structure.

Software Development Methodology

The first step in developing communication between the software system and the imaging hardware is to develop control classes in Visual Basic.NET for each of the hardware controllers' respective APIs. The controllers used for the development of this system all communicate with the computer via a unique set of ASCII text commands. These commands were converted into Visual Basic functions, with the output of each function being the ASCII text required to execute that certain command. Due to variations in instruction sets between controller manufacturers, different control classes had to be developed for each manufacturer used. In the case of motor controllers, in order to provide a layer of abstraction to the individual libraries of each motor, all of the motor controller classes were superclassed to an abstract **Motor** base class. This **Motor** class served as the subject of all commands sent by the GUI. It also abstracts the connection protocols between hardware and computer. For instance, several motor controllers connect to the computer via Ethernet with a static IP, while others connect via a serial connection. However, due to this class structure, the software system can implement commands for any type of motor, given that the new motor is a subclass of **Motor**. This means that if a user wants to add a new manufacturer of motor hardware to the system, they must simply adapt the hardware's library of commands to the system as a new motor object, and the system will understand it. This feature was designed to decrease the cost of developing a new software system for every new hardware system that is introduced.

In order to achieve full customization of various different hardware components within the system, the instantiation of specific control objects were required to be detached from the control of those objects. The factory design pattern, a commonly used software design pattern, was used in order to achieve this abstraction between connection and control [3]. When a given GUI control element is generated on the control screen, that element calls upon its factory method to determine the parameters by which to "construct" a control object for a given piece of hardware. For instance, upon creation of the "Advanced" control screen, each GUI element instantiates a new **MotorFactory** object. This **MotorFactory** determines the type of motor object to be created based upon the options selected by the user during the configuration period, and then instantiates that control object, linking its commands to the specific GUI element it was created from. A description of the software class structure can be seen in Figure 1.

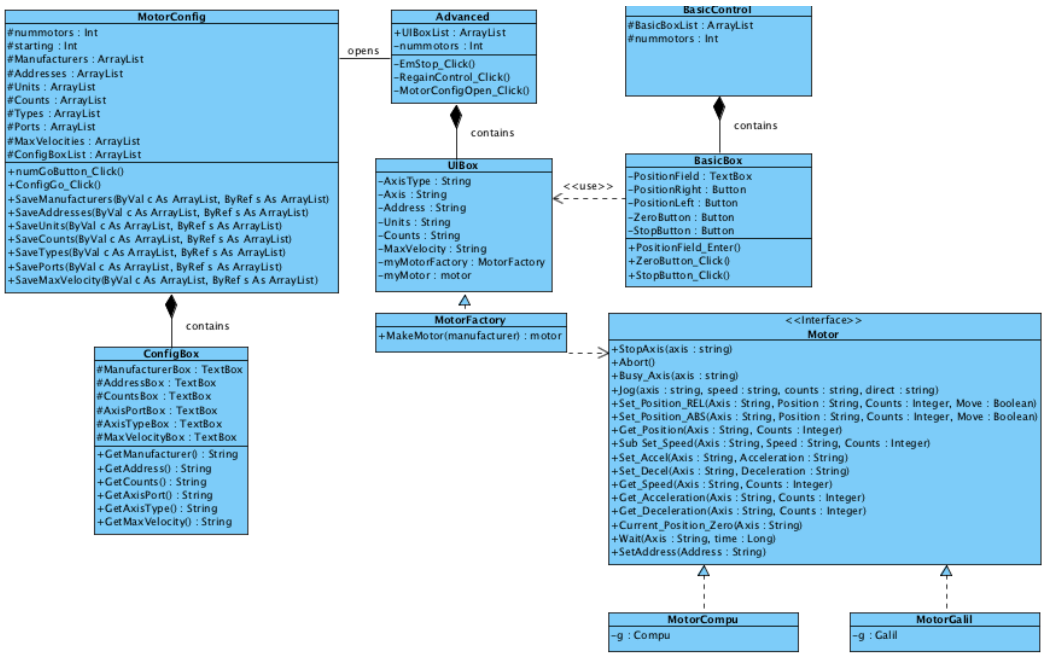

Figure 1: UML Class Diagram

GUI Design

One of the main goals of the project was to develop a dynamic user interface that, at runtime, could resize based upon the quantity and types of hardware connected. This goal was ultimately achieved using a basic object-oriented approach. Each motor setup panel on the configuration screen is encapsulated as its own **ConfigBox** object. Based on the user selection of motor quantity, the program instantiates the appropriate number of **ConfigBox** GUI elements on screen, resizing the window appropriately. The **ConfigBox** panel contains various parameters for motor control, including motor manufacturer, IP address/serial port, measurement units, and axis type. All data entered into each **ConfigBox** is saved to arrays of each control parameter.

The **Advanced** control window contains all of the basic functions required for basic control over the stage motors, x-ray source, and detector. These essential functions include stage rotation, autojog start/stop, velocity control, and image acquisition. In addition, an emergency stop button appears on every screen of the GUI, to turn off all motors and radiation sources. Each of the sliders are not directly linked to the hardware controllers – instead, they are each linked to a respective control window on the advanced control screen. The advanced window contains finer control over the system, with settings to change the acceleration, deceleration, jog speed, and direction of the rotary axis. This window also contains an emergency stop, with a controls lockout until reset by a user. Each GUI element is dynamically added to its respective control window based upon the user settings. This allows for an efficient use of screen space as well as flexibility to add more control if a given system requires it. Through this method, the GUI can automatically accommodate multiple motors and hardware systems from a single window.

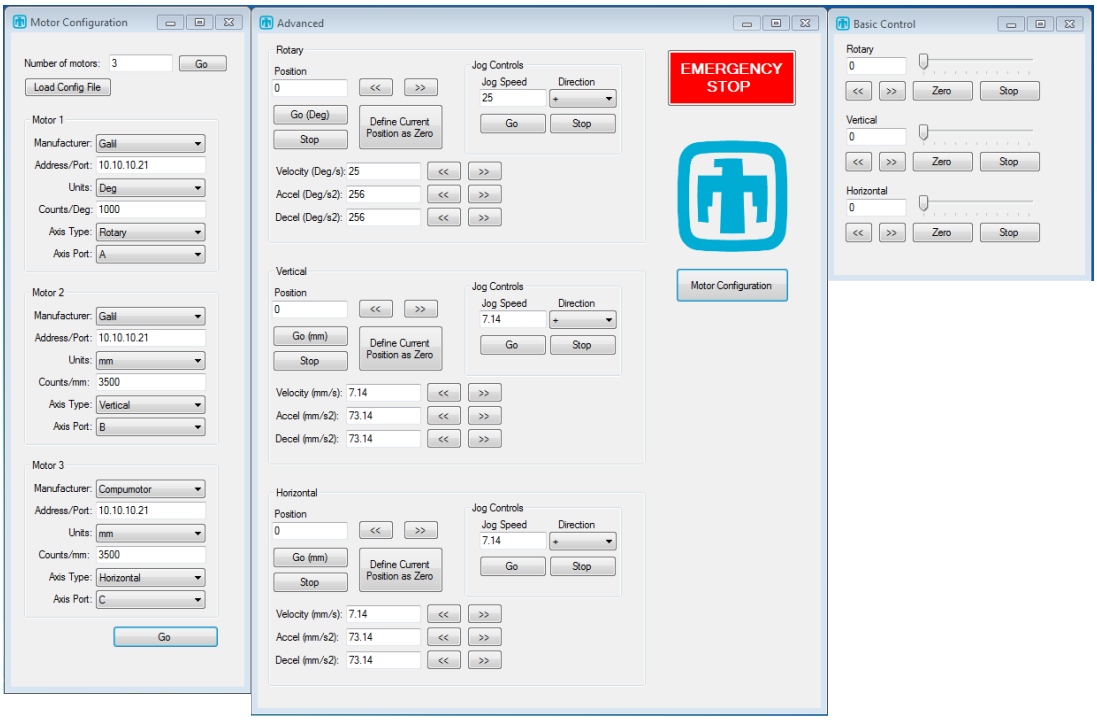

Figure 2: Software GUI Layout

RESULTS

When the software is initially launched, users are presented with a configuration screen, as shown in Figure 2. Upon entering configuration data, the configuration page is automatically hidden and the basic control window is displayed. This window (as seen on the right in Figure 2) is used for a majority of hardware control on a daily basis. If a user at any point requires finer control over the hardware system, they can simply open the **Advanced** control screen via a

button on the basic control screen. Just like the configuration window, all controls on the advanced and basic control windows are dynamically generated at runtime. Because of the abstraction of the hardware APIs to generic hardware classes, each **UIBox** panel on the screen is linked to a generic hardware object. As a result, users can easily add new hardware APIs to the software. This process is as simple as adding each function of the API from the motor controller and then adopting the function names that were used in the Motor Class. This method was tested with two motor manufacturers. Two motor controllers were connected to the system through two distinct communication interfaces. Upon designating these controller manufacturers and communication interfaces in the configuration window, the software was capable of independently controlling the two motors connected to each of the two motor controllers with all required functionality.

CONCLUSION

Upon completion, DRASS will allow for many different configurations for CT and X-Ray imaging systems with little-to-no code adjustments. The implications of this software are two-fold. Firstly, having software that can adapt to multiple motor controllers from different manufacturers can result in reduced costs due to fewer software licenses and a lower need for training. Currently, each controller from a unique manufacturer requires its own piece of specialized software to establish communication. Each copy of the software can cost up to tens of thousands of dollars. With the implementation of DRASS, organizations will only be required to purchase one piece of software that is capable of completing all required tasks. In addition, researchers will only be required to learn one piece of software that is capable of all required functionality, with minimal training. The second implication of this software is that its flexibility allows for far more complex hardware configurations. Because hardware systems can be dynamically added and removed from the software at runtime, researchers now have the freedom to design CT and X-Ray scanning systems in new uncommon configurations for use in specific applications. With successful implementation, this software will potentially save Sandia National Laboratories hundreds of thousands of dollars, and will allow the organization the freedom to further its computed tomography research in ways that were formerly difficult or impossible to execute.

ACKNOWLEDGMENTS

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

REFERENCES

1. Wang, S., Huajie Han, Kun Gao, Zhili Wang, Can Zhang, Meng Yang, Zhao Wu, Augusto Marcelli, and Ziyu Wu, " A LabVIEW based user-friendly X-Ray phase-contrast imaging software platform," Journal of X-Ray Science and Technology, vol. 23, no.2,pp 189-199, 2015
2. Ortega G.A., "Desarollo de software de librerias y aplicaciones para el control de sistemas micro-CT," M.S. thesis, Dep. Bioengineering and Aerospace Eng., Universida Carlos III Madrid, Madrid, Spain, 2013.
3. Gamma, E., Richard Helm, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, first edition. Addison-Wesley Professional, Reading, MA, 1994.