You must provide a software design document for your project. This document must include, as appropriate, the following design viewpoints as documented in IEEE 1016-2009: Context; Composition; and, Logical.

Please note that this means your document should include:

- An overall description of the purpose and objective of your finished product, including a description, when appropriate, of the problem or gap it is trying to fill;
- A description of the users;
- A description of the use cases;

Identification of the major components of your software:

- For each major component, the function and non-functional requirements of that component;

Identification of the classes, interfaces, and similar software modules you will use to implement the components:

- Where helpful, control flow graphs and data/information flow diagrams to highlight interactions between components or between users and components

Moreover, this should include documentation, to include graphical mock-ups or command-line documentation, of the user interface portion of the Interface viewpoint, but does not need to provide documentation on the internal interfaces that are not intended to be used by the final customer.

Finally, for each logical software module you identify, you must provide a timeline indicating when you expect to begin and when you expect to complete development, testing, and integration.

5.2 Context: Design Viewpoint

IDEF0,
UML use case diagram,
Structured Analysis context diagram

5.3 Composition: Design Viewpoint

Logical: UML package diagram,
UML component diagram,
Architecture Description Languages,
IDEF0, Structure chart, HIPO
Physical: UML deployment diagram

5.4 Logical: Design Viewpoint

UML class diagram,
UML object diagram

- An overall description of the purpose and objective of your finished product, including a description, when appropriate, of the problem or gap it is trying to fill;

Joseph Crandall and Karl Prisner are collaborating to deliver a Biological Robotic Imaging Experimentation Framework (BRIEF). In this case framework means a physical stand and robotic arm accompanied by point cloud interpretation software and ROS driven robotics with the aim of creating biological sample development movies over time while manipulating the samples as needed. By working with Dr. Simha and the GW Biology department we have identified that they have a need for a device with these capabilities to conduct research.

Our finished project will include a robotic arm, imaging stage, rotating point cloud sensor and a robotic gripper. Our robotic systems will be controlled through the Robotic Operating System (ROS). Our system will be simulated in a Gazebo simulation world.

Our framework will provide a researcher with quantitative point cloud data of the changes of the geometry of a biological process over an extended duration of time. This capability is currently not available to the George Washington University Biology Department.

- A description of the users

A biological researcher could use our product to conduct an experiment where a plant's development over time could be monitored and then played back through a point cloud film where the geometric changes could be viewed from all angles. CUrrently we do not have a user interface planned since the control of the robot is run through a linux command line interface by issuing ROS commands. Once the development has reached a point where a user can issue a command to the robot and to the simulator and they behave identically we will look for ways to implement a mouse friendly interface. At this time no work has gone into this component of the research.

- A description of a use case could be

A typical use case involves a biological sample, BRIEF, and a researcher. The goal is to be able to minimize the amount BREIF system knowledge the researcher has to have so that they can easily use or system to acquire data. The end goal is to have a washing machine type experience. Place your sample, designate the timescale of the experiment and what you would like to monitor and then begin. We are a ways from this at the moment.

Identification of the major components of your software:

- For each major component, the function and non-functional requirements of that component;

Due to the scale of this project, Joseph Crandall and Karl Prisner have split to workload of the project so that we may better tackle components that will eventually be brought together in the final system. Joseph Crandall will be responsible for understanding how the ROS operating system works with the Schunk Robotic Arm, and will also have to figure out how the open source yale grabber will be ported from its current python code so that it can work in unison with the Schunk Robotic arm in a ROS based environment. Behind the robotic operating system that will drive the robots will be a series of launch files that issue the joint movement. As the team becomes more experienced at writing launch files will will start to work on improving the usability of the system so that novices can move the robot without having to understand the details of ROS. Will working on moving the Robot, Joseph Crandall will also continue his work on a simulator or the robotic system. The simulator being used is Gazebo and it is important for several reasons. Since there are multiple mechanics components for the project including the arm, the grabber, and Karl Prisner's Imaging arm, it is important to understand how all the components work in unison. The Gazebo simulation will allow our team to learn how to execute commands in both a simulated environment as well as the physical one. From an inverse kinematics perspective this is necessary because if the timing of actuators are

not properly tested there is a possibility that the robotic arm and grabber will collide with the imaging arm. The ability to import point cloud data into the simulated world is one way that we may be able to correctly manipulate the point cloud data, representing what is on the stand, with the robotic arm and grabber.

- For each major component, the function and non-functional requirements of that component;

An outline of the packages that have been explored for the project is explored below.

Schunk LWA 4P driven by schunk_canopen_driver on ros

The Schunk Light Weight arm is where most of Joseph Crandalls work had been dedicated. A majority of the allotted project time has been spent learning about ROS (the Robotic Operating System) in order to properly use the package schunk_canopen_driver in order to send commands to the robot to perform actions. The initial functional requirements of this system focus on having a full understanding of what is already written within the schunk_canopen_driver and the ability to execute all possible user directed commands. Once this is complete the second set of functional requirements will focus on giving the arm point cloud data of the object that has been imaged. The driver for the arm will then have to decided how to send a series of movement commands to the arm to interact with the object represented by the point cloud.

Yale Open Hard driven by python script

The first functional requirement for Yale Open hand is to be able to run the python script to properly open and close the hand. The second functional requirement is to create a ros package for the yale open hand so that it can be driven within a ros environment. The third functional requirement is to incorporate the ros open hand package with the schunk package as well as physically attach the open hand to the robotic arm. The goal is to be able to calculate inverse kinematics on the full system so that the arm and hand can be driven as a single robotic system.

Gazebo Simulation

The Gazebo simulation world allows for the testing of robotic components, integration of multiple components and how commands are sent. The simulation acts a test bed where launch files can be executed without fear of damaging the robot due to improper movement of joints that may cause the robot to collide with itself or with another object in the world. The simulation also proviudes a visualization for all of the components of the project.

Imaging Stage, Xbox Kinect, & Imaging Arm

These components are be developed by karl. Once they are working independently we will combined the arm and grabber in the Gazebo World simulation. At this point the arm and grabber will be mounted in the center of the imaging state where it will be able to reach any location on the stage. At this point a new Ros package will need to be written to incorporate the movement of the imaging arm with the movement of the schunk arm and yale grabber to be able to calculate inverse kinematics on the whole system. With this many components moving, the possibility of collision exists so it is important to have all components run from the same driver.

Identification of the classes, interfaces, and similar software modules you will use to implement the components:

- Where helpful, control flow graphs and data/information flow diagrams to highlight interactions between components or between users and components

Moreover, this should include documentation, to include graphical mock-ups or command-line documentation, of the user interface portion of the Interface viewpoint, but does not need to provide documentation on the internal interfaces that are not intended to be used by the final customer.

Packages

At this state in the project is is better to think in terms of package because this seems to be the method in which ROS designates a specific set of functions for a robot or ROS control system. We already have the Schunk_canopen_driver package. This page will need to be extended in order to add a form of point cloud interpretation and interaction.

We will need to write a new ros package for the Yale Open Hand which currently manipulated through a series of python scripts. We will also have to create a model of the hand for the Gazebo simulation.

Interfaces

Currently the only way to interact with the ROS code or the Gazebo simulation is though an Ubuntu Linux command line interface. As the project matures it will be important to identify a more user frields method of running experiments with the system. At this time however this is not a priority since we do not fully understand how all the robotic systems, command files, and simulation will interact. Once we have a better idea about how these systems interact we will be able to think about an interface that provide the use with the content and control they want without having to deal with all the minute details associated with ROS

Software & Hardware Development Timeline

Dec 2016 - January 2016

- Complete Gazebo Model of the Schunk LWA-4P and issue launch files to manipulate the Gazebo Simulation
- Synchronize commands and manipulation of gazebo simulation with physical Schunk LWA 4P

February 2016

- Operate Yale Open Hand with Python Scripts
- Create ROS Package for ROS based Open Hand
- Create Gazebo Simulation for ROS based Openhand

March 2016

- Integrate Schunk Lightweight arm and Yale Open hand in a single ROS package and create a Gazebo World of the combined system.

April 2016

- Integrate arm and hand with Karl's imaging stage, imaging arm, and xbox kinect
- Create new ROS package and Gazebo world simulation, focusing on incorporating the point cloud data.

May 2016

- Start work on user interface for the system.

Coments

This is a very ambitious schedule and from past experience working with ROS and Gazebo, steep learning curves can present themselves as progress is made. Each one of these needs to be internalized before moving forward. Dr. Simha has stated that this research project will continue on after Joseph Crandall and Karl Prisner have graduated so it is our goal to make our work as transparent and easy to pick up for the next students who join the research group. We have put a large amount of work into understanding the foundational material for ROS as well as fabricating parts for Karl's component.