**Basic Information:**

Why use multiple different processes?

* We need at least 2 different processes because we are using two different CPU modules. Each CPU needs a process to run (can be the same or different process).

Why don’t we just use two processes?

* Separating our software’s by functionality into different processes leads to greater encapsulation (Camera Control doesn’t need to know image processing stuff) and organization. It also allows for more people to work in parallel at developing this architecture because the processes are independent of each other (minus communication).

Your software map looks so complicated and there are so many messages. I don’t think we can build something like this????

* We will use an object oriented language in order to abstract the functionality of our code in order to make it reusable. Our main processes (GNC, S/C Comms, WatchDog, ect…) will be sharing a library of lower level classes, which we will develop based on the functionality needs of each process. These lower level objects will serve to complete a specific function and will be used by every process in order to minimize repeated code. Examples of these lower level objects will be a selector object (Used for Event Driven Servers), and a message handling object (which will encapsulate all of our process communication).

**All Processes:**

* All processes will communicate over TCP/IP non blocking sockets
* All processes will attempt to fix themselves when they become sick. If they cannot fix themselves, they will exit and be restarted.
* All processes will be using timeouts for the processes that they request actions from

**WatchDog:**

Input Messages:

* Process Health and Status Response

Output Messages:

* OSPRE Status
* Process Health and Status Request

Functionality:

* WatchDog will monitor the health and status of all processes
* WatchDog will be capable of restarting processes that are unresponsive
* WatchDog will start every process on initialization
* If WatchDog fails, it will be restarted by the initialization script
* WatchDog will create the OSPRE Status message that gets transmitted to the S/C
* WatchDog will monitor any package health sensors in the OSPRE sensor package
* WatchDog will take Process Health and Status Response message from every process and sensor data to decide OSPRE health and status

Notes:

WatchDog will have a unicast socket for receiving Process Health and Status Response Messages. WatchDog will send all other responses on direct socket TCP/IP connections. (NO MULTICAST)

**S/C Comms:**

Input Messages:

* OSPRE Status
* Data Request
* Pointing Request
* Solution Message
* Spacecraft Data Message
* Spacecraft Status

Output Messages:

* OSPRE Status
* Data Request
* Pointing Request
* Solution Message
* Spacecraft Data Message
* Spacecraft Status

Functionality:

* S/C Comms will relay all messages to and from the S/C.

Notes:

The S/C Comms will be the only software module communicating with the S/C. The S/C Communicator module insulates the OSPRE system from the S/C. If the S/C changes, the S/C communicator is the only thing that needs to change.

**Camera Control:**

Input Messages:

* Spacecraft Status Message
* Image Adjustment Message

Output Messages:

* Data Request
* Image Message

Functionality:

* Camera Control will be responsible for all communications with the camera
* Camera Control will be able to request pictures from the camera
* Camera Control will be able to change the settings of the camera
* Camera Control will have the logic on how to best change the camera settings based off of Image Processing Image Adjustment Message
* Camera Control will decide if it can take a picture based off of its settings and the S/C State Message. If Camera Control is able to capture an image, it always will capture the image.
* Camera Control will mark each picture that it takes with a Request ID and Time Stamp
* Upon image capture (some delay here), Camera Control will send Image Message to Image Processing and a data request message to S/C Comms

Notes:

The Camera Control will be the only software module communicating with the camera. Therefore, the Camera Control module insulates the OSPRE system from the specific protocols and implementation details of the specific camera utilized. The Camera Control module will be the only module that knows the settings of the camera, the cameras settings range, and cameras capability.

Camera Control will decide if it can take a picture based off of the exposure settings and the angular velocity of the S/C, what the S/C is pointing at and other possible inputs.

The Camera Control generates the Data Request Message because it knows the most accurate time that the picture was taken. To reduce error, we would like the Spacecraft Data to match the time that the picture was taken. We have to determine how accurate our clocks are (deci, centi, millisecond) and determine how accurate we need them to be. The Camera Control and S/C might need to sync clocks every so often.

**Image Processing:**

Input Messages:

* Image Message

Output Messages:

* Image Adjustment Message
* Processed Image Message

Functionality:

* Image Processing will grab the important numbers from each Image and pass them to GNC (Angular Diameter, alpha, beta)
* Image Processing will forward the Request ID number of each Image, and what the Image is of to GNC
* Image Processing will be able to tell the pixel error associated with each measurement it provides to GNC
* Image Processing will destroy photos after it is done with them in order to save memory
* Image Processing will diagnose how the images can be improved and send a Image Adjustment Message to camera control.

Notes:

Anthony will take lead on this.

The logic behind how to adjust the camera will be in Camera Control because it is important that only Camera Control know the details of the camera. If we were to put that logic in here, Image Processing would have to know the ranges of what settings were possible of the Camera which would break encapsulation (BAD!!) .

**GNC:**

Input Messages:

* Processed Image Message
* Spacecraft Data Message

Output Messages:

* Pointing Request
* Solution Message

Functionality:

* GNC will calculate solutions
* GNC will calculate the error in the solutions
* GNC will send Solution Messages to S/C Comms when the error in the solution is acceptable
* GNC will determine what object needs to be imaged in order to meet the error requirements.
* GNC will create Pointing Requests for the S/C

Notes:

Cameron will take lead on this.

**S/C:**

Input Messages:

* OSPRE Status
* Data Request
* Solution Message
* Pointing Request

Output Messages:

* Spacecraft Data Message
* Spacecraft Status

Functionality:

* Power on the OSPRE sensor package
* Command OSPRE to switch states from Active to Passive and from Passive to Active
* Point the sensor package to where it requests (Hardware)
* Request current solution despite error

Assumptions:

* S/C will respond to every message that gets sent to it

Notes:

Either has to tell OSPRE that it can or cannot take pictures OR needs to pass the angular velocity of S/C to OSPRE.