Requirements Specification

Prepared by: The Skyentists  
Due Date: 10/03/19  
Updated: 11/07/2019

Changes for Version 2

1. Gave context to the questions asked to the users in User Feedback section
2. Provided more in-depth explanation for some of the functional requirements
3. Added better phrasing and some examples to some of the non-functional requirements
4. All team member signatures were added
5. All ethics certificates for the team members were added
6. Added medium priority requirements based on client feedback

# User Overview

The Client:

NASA expects the client to provide the SMAP L4C data product (SMLAP L4C). At a high level, the client would need to read in real world plant and soil data, calibrate the modeling software to match the ground truth data, and then simulate the SMAP L4C based off these optimized parameters.

Other NTSG employees:

If the client didn’t have the want or need to produce the SMAP L4C, then other employees at NTSG can produce the data product. They would have to learn how the calibration process works and go through all the tasks that the client would perform.

Future Climatologists:

If NTSG were to shut down and climatologists still wanted the SMAP L4C, they would have to perform all the tasks other NTSG employees would perform but with less technical expertise.

# Hierarchy [or Organization Chart] of Users

As illustrated in Figure 1, climatologists may use the output of the calibration process to further improve other more complex climate change models. Most of the data is provided by NASA, who funds the Numerical Terradynamic Simulation Group (NTSG). Currently the client, an employee for this company, oversees calibrating NTSG’s climate change model. If the client is unable to perform this calibration, then other NTSG employees will use this software.

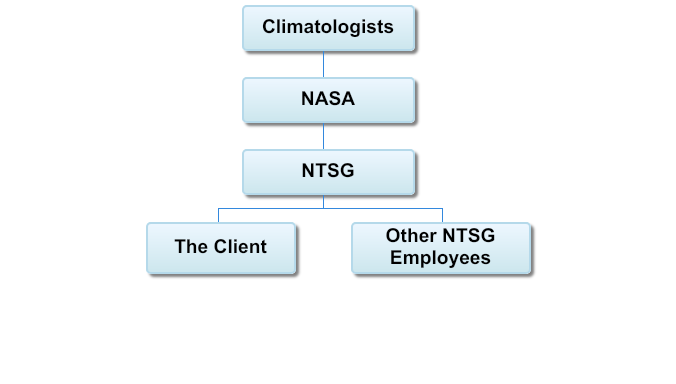


Figure 1. Hierarchy Chart

# User Groups

Table 1.

This table represents the User Groups that will be using this software system. The first column represents the name of the user group, the second column represents the number of users that belong to said group, and the third column describes the details of how the User Group will be using the software system. The details revolve around what permissions they have, what they do at their current business, what features they will be using of the software system, how often they will be using the software system, what hardware/OS they will be using the software system on, their language, and if they will be interreacting with the system directly or indirectly.

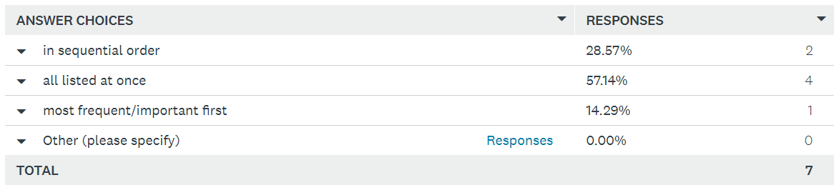
|  |  |  |
| --- | --- | --- |
| **Name** | **Number** | **Description** |
| The Client | 1 | The client will be the owner of the GIThub repository for the project, so he will control the repository settings and have administrative control over what gets pulled and pushed to the repository. The clients main work task is Earth System Model Calibration, so he will be using all of the features available from this software, which include the library of data structures and functions, The Flux Tower Observational Record Compiler, The PFT Selector and Calibrator, The Pre-Optimizer and Optimizer for GPP, The Pre-Optimizer and Optimizer for RECO, The SOC Parameter Fitter, The Analytical Model Spin-up, and The Numerical Model Spin-up (Approximation of features, subject to change later in development). This software will be used by the client daily. The Client appears to have a vast knowledge of Computer Systems and Application Domains. The Client will most likely being using a Linux machine, though could also be using other UN\*X machines as well. The Client speaks English. The Client will be interacting with the system directly. |
| NTSG Employees | Approximately 20 (Based on NTSG website) | The NTSG Employees will most likely be able to push and pull from the GIThub repository but will not be able to make changes to the repository settings. The NTSG employees work in the research of ecosystems using satellite information, so they will mostly be using the data structures and functions of the software rather than the software itself, which would probably vary between daily and weekly. However, they will be using the software bi-yearly for the calibration of the main model. The NTSG Employees will most likely have a moderate to high knowledge of computer systems and application domains. The employees will be using UN\*X machines due to the client specifying that the software is going to be built mainly for those machines in mind. Considering NTSG exists in UM, it is safe to say they are all English speakers. Interactions will vary dependent on what each employee is doing, so could be either direct or indirect. |
| Other Scientists | 1 Million+ | Other scientists will be able to pull from the repository, but will not be able to make changes, so if they want their own version, they will most likely be forking it too their own accounts. The tasks they preform during their business hours will vary. Considering how broad this group is, they could be using both the software and all its features along with the data structures and functions library of the software, but they will most likely on average be using the software only for special cases, so maybe every 5 years for the average scientist, but find more utility out of the data structures and functions library, which they will be using weekly at least if they have the need to use it. Computer system and application knowledge will most likely vary, averaging probably between moderate and high knowledge. Despite the system being made purely for UN\*X machines, people will probably find a way to use it on other OSs such as Windows. Since this system is made in America, it will more likely have more American users, so the majority of scientists will be English speakers, with a minority of scientists of other languages. Language shouldn’t be a problem, however, since this is an open source projects, so different versions in different languages can be created by others. They will most likely have a more indirect interaction with the system. |

# User Feedback

The method we decided to use for user feedback was a 4 question survey that was given to 7 of our DBS and CS (with modeling experience) peers. Here are the results of the survey:

Question 1: How would you prefer to select a certain category to edit?

This question pertains to the calibration process, because the user will have to specify which plant function type (PFT) they are currently calibrating. There are nine total PFTs for this process so the user would choose the relevant PFT eight times.

Result:

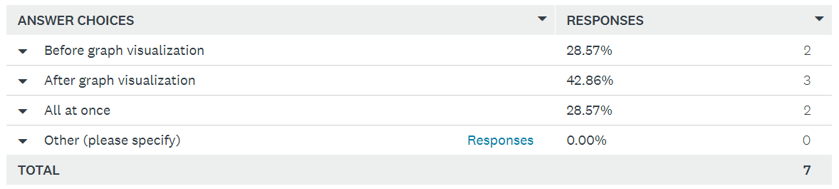
Summary: It seems that the majority of the responders said that they would prefer that all the choices be listed at once when selecting a certain category with a 57.14% vote. The follow up would be that people would want to select parameters in sequential order with a 28.57% vote. Only one person wants to select categories based on frequency and importance (14.29%). No one had any other options they wanted to specify.

Question 2: For editing the parameters for a model, would you rather edit each….

Result: 

Summary: The majority of the responders say they would prefer to edit parameters one by one as opposed to doing it all at once and grouping certain parameters together and then editing them. The second option people would settle for is “at once in one big grouping” with 28.57%. Only 1 person preferred editing in groups of certain parameters (14.29%).

Question 3: When would you prefer to edit parameters?

Result: 

Summary: There was a more even split on when the responders want to edit parameters. The highest percent goes to doing it after the graph visualization, though the two other options, before graph visualization and all at once, are close behind with the same percent. It seems that there is a little more subjectivity on when people want to edit parameters. However, no one had their own ideas on when to edit parameters.

Question 4: Would you like to see what parameters are being used to calculate?

The user would not have any control over the editing of the values for the relevant parameters, but only control which parameters are being optimized with the model data.

Result: 

Summary: All responders want to see the parameters that are being used to calculate with a perfect 100% on yes.

# Functional Requirements

Functional requirements are the stipulations of how the software will behave. The functional requirements that the one of interest is dependent on is denoted inside the brackets () at the end.

High Priority:

The following list displays the high priority functional requirements, or the most critical behaviors of the software. These range from directly affecting important business processes to improving workflow. These requirements will have, at most, one other high priority dependency in the software and will have lower priority dependencies.

The software shall:

1. Calculate dominant Plant Functional Type (PFT: Evergreen Needleleaf, Evergreen Broadleaf, Deciduous Needleleaf, Deciduous Broadleaf, Shrub, Grass, Cereal Crop, and Broadleaf Crop)for each tower site
2. Read in 4 input datasets (L4C reference dataset, L4C meteorological input dataset, Flux Tower Fluxes dataset, IGBP FAO Soil Organic Carbon Inventory dataset) for calibration
3. Compile the L4C meteorological inputs provided from NASA/NTSG and ground-truth data from flux tower sites into table
4. Allow the user to choose one of the nine PFTs (High-1)
5. Guide the user through removing outliers in average annual GPP and RECO calculations
6. Allow user to choose which parameters to use when optimizing GPP and RECO, such as LUE (light use efficiency) for GPP or faut (autotrophic/plant respiration fraction) for RECO
7. Calculate linear ramp functions given current BLPUT (High-6)
8. Allow user to specify number of Numerical Spin-Up iterations
9. Compute comprehensive validation and fit statistics: graph flux tower data against model-estimated data
10. Output updated BLPUT, 4 SOC stock-size maps, and SMRZ minimum and maximum for 2000-2018
11. Calculate flux tower weights as some tower sites are located in same 9km section

Medium Priority:

The subsequent list shows the medium priority functional requirements. These are the software behaviors that directly or indirectly affect business flows. These requirements will be dependent on the high priority requirements and may have some low priority dependencies. They are still vital in the calibration process itself, but are dependent on the high priority requirements

The software shall:

1. Calculate percentage of area that is occupied for each PFT at each tower site
2. Compile historical data and error metrics for each day for each PFT into table
3. Subset time series variables to user configurable period (High-2)
4. Subset L4C meteorological and L4C reference input to sites that have dominant selected PFT (High-4)
5. Compile ancillary info on each site for selected PFT such as dominant PFT at 9km scale and SOC stock size for the site (High-4)
6. Compile all important data for calibration of selected PFT that includes FPAR, PAR, TSURF, TSOIL, SMSF, VPD, TMIN, SMRZ, minimum SMRZ, maximum SMRZ, and PAW (High-4)
7. Average data for PFT each day of the year (High-5)
8. Display current ramp functions to the user and allow them to save the plots as files (High-7)
9. Allow the user to plot GPP against Emult separately form the ramp functions (Medium-8)
10. Report the differences between new and old value parameters after optimization (High-7)
11. Allow the user to specify Pk and Prh for RECO (High-6)
12. Create 2 plots: Rh/Cbar against TSOIL with the Arrhenius curve on top, Rh/Cbar against SMSF with the SMSF ramp function on top (High-7)
13. Calculate Cbar for each tower site after optimization (High-7, Medium-12)
14. Calculate σ and Bsoc for each tower site to plot σ \* Bsoc against ground truth SOC sizes
15. Run the Analytical and Numerical Model Spin-Ups (High-8)
16. Run preliminary spin up arbitrary period over full operational record 2000-2019
17. Run L4C soil model forward runs
18. Prepare vectors of initial optimized parameters and GPP and RECO optimization objective functions to iteratively change optimization parameters
19. Calculate GPP\*, Kmult\*, NPP\* based off of calculations (Medium-7)
20. Calculate RMSE for each subset of towers for observed both GPP and RECO against L4C Reference Dataset, optimized GPP/RECO, and L4C Forward Model Run (High-9)
21. Calculate NEE (NEE= RECO – GPP) based on optimized parameters then report stats of NEE against L4C Reference Dataset, optimized GPP/RECO, and L4C Forward Model Run

Low Priority:

The following list is the low priority functional requirements, which will be completed depending on time and the completion of the above two lists.

The software shall:

1. Remove negative values in annual GPP and RECO for each flux tower site (Medium-7)
2. Calculate lower and upper limits for APAR (Medium-8)
3. Allow the user to repeat optimizations after the initial optimization (High-6, Medium-10)
4. Allow the user to plot GPP against Emult and Rh/Cbar against Kmult

# Non-functional Requirements

Non-functional requirements describe how the software should behave and how well the operator is able to use the program. All requirements listed below are numbered and shall be referenced by the first letter of their priority followed by their index (e.g. the first high priority requirement is H1).

High Priority:

The requirements listed here are necessary for a minimum viable product. The project is considered a failure with the absence of any high priority requirement.

The Software shall:

1. Have accuracy of calibration that meets or exceeds the current implementation.
2. Be able to process data while maintaining the size being used in the current implementation.
3. Pass a holistic test suite (a script that individually and automatically tests the functionality of the entire project, such as Python’s built-in “unittest”).
4. Be subjectively rated better than the current implementation by the client.
5. Follow a modular, object-oriented design pattern.
6. Have basic documentation for all features.

Medium Priority:

These requirements, while not integral to the system, will improve various aspects of the software.

The Software shall:

1. Have well-written inline documentation for every class, method, and variable (the code itself).
2. Allow the process, as a whole, to run 20% faster than the current implementation.
3. Use 30% less total, saved memory than the current implementation.
4. Make use of Python virtual environments.
5. Have a test suite for every component.
6. Have sufficient maintainability as determined by the client.

Low Priority:

Requirements in this list have little to no impact on how the system will behave but add value to the project.

The Software shall:

1. Have a test case written for every method.
2. Have a test suite with 95% code coverage or higher.
3. Be validated on systems other than those specified in the software requirements.
4. Be validated on browsers other than those specified in the software requirements.

# Hardware Requirements

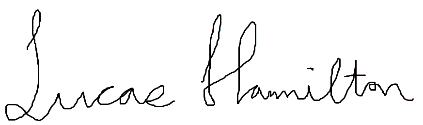
The Software shall be developed on x86\_64 architecture with SSE4 extensions or later.

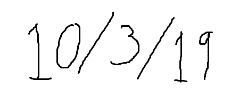
# Software Requirements

* The Software shall be written in Python 3.
* The Software shall run on Unix and GNU/Linux based systems.
* The GUI shall run in a web browser (Mozilla Firefox or Google Chrome).

# Execution and Acknowledgement

The team members hereby indicate by their signatures below that they have read and agree with the specifications of this document.







|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| [Insert name & date here] |  | [Insert name & date here] |  | [Insert name & date here] |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| [Insert name & date here] |  | [Insert name & date here] |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| [Insert Client name & date here] |  | [Insert Client name & date here] |  | [Insert Client name & date here] |