Test Plan

Instructions: Pick one testing strategy from each of the following two categories that your team will employ as you implement your software. Note that the suggested methods are in bold and underlined. If your team chooses either Load testing or Integration testing, meet with me to discuss appropriate test plan format in places where it might differ from that below.

1. **Unit testing** OR Code review
2. **Functional testing** OR Load testing OR Integration testing

Document your test plan using the template provided below.

# The Skyentists: GPP and RECO Calibration Software

Mark Matas, Lucas Hamilton, Jake Pennington, Max Thibeau

# Date:

04/17/20

# OVERVIEW

## Purpose of the Test Plan Document

The Test Plan defines the testing strategies and approaches for the completion of testing on the calibration software. This document also includes the results of the testing scenarios. The intended viewers of this test plan are the developers of the calibration software and any future developers and testers, although this document can be shared with any user/client/stakeholder whose input is needed and valued.

## Testing Strategies

The team’s overall approach to testing includes both the unit and functional testing strategies. Unit testing is an approach that includes testing an individual object or function with a single test. Testing in this way allows the tester to set a specific input with an expected value, as unit testing tests a specific behavior, function, or object in the code. This approach includes the testing of all steps within software development that consists of architecture and design, code implementation and debugging, performance, and quality assurance (1). This unit testing blends well with the functional testing strategy since some of these unit tests could be included in functional tests and also can test many of the same parts of the software. Functional testing is an approach that consists of a series of tests that define entry values for an operation and observe if the result is what was expected (2). These tests can both be used to ensure the right error message displays or if the expected result is a value. This type of testing also allows for normality tests, as well as exceptions.

## Test Risks / Issues

[Describe the risks associated with product testing. Also outline appropriate mitigation strategies and contingency plans for such risks.]

# 2. TESTING METHOD 1: UNIT TESTING

## 2.1 Scope

The functions that will be tested are the calculations within the software and the inputs prompted to the user. The items that are not tested are the functions within the graphical user interface (GUI) and the getters/setters within the classes, as these will work properly. These topics are further described in the table below, with a reason why it is tested or not. The tester for the functions is also reported.

|  |  |  |
| --- | --- | --- |
| Unit Test Scope |  |  |
| **In Scope** | **Reason Tested** | **Tester** |
| *List functions that will be tested.* |  |  |
|  |  |  |
|  |  |  |
| **Out of Scope** | **Reason Not Tested** |  |
| *List functions that will NOT be tested, and give a brief explanation as to why.* |  |  |
| Getters/Setters | These functions are built into python and used in most/all classes in the project. Someone can test only one of these and it would work for all the other, if necessary. |  |
| GUI Functions | These are the GUI functions that make the pages flow and get to the base page. These are the functions that were previously tested in the prototype design phase in the first semester (Fall 2019). These GUI Functions include the “display” functions within the CLI (can test by making sure x and y will be same length) |  |

## 2.2 Methodology

[Describe the testing procedure to be used to test the project’s product. Provide an outline and brief description of all testing steps and who will be responsible for each. Use the resources discussed in class and posted on Moodle to help you complete this section and to provide additional templates. Be thorough!

If you are using code review, be sure to include a clearly labeled and well-articulated section on the standards and guidelines your team will use during the review process. Also describe the rules and processes your team will follow when dealing with results of the code review. ]

## Test Readiness

[Describe what conditions have to be met before the testing procedure can commence.]

## Control Procedures

### 2.4.1 Problem Reporting

[Document the procedures to follow when an incident is encountered during the testing process. It is highly recommended that a standard form is used for this; attach a blank copy as an "Appendix" to the Test Plan. In the event you are using an automated incident logging system, write those procedures in this section.]

### 2.4.2 Change Requests

[Document the process of modifications to the software. Identify who will sign off on the changes and what would be the criteria for including the changes to the current product. If the changes will affect existing programs, these modules need to be identified.]

## Test Cases

Table 1: Unit Test Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Unit Test Cases | | | |  |  |  |  |
| **ID** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Tester Name & Date** | **Pass/Fail** | **Resolution** |
| 1.0 | ConfigFile Function: test\_read() | Relative path for configfile | Result = prints out valid object |  | *Jane Doe, 2-15-2019* | *Fail* | *Testing incident report #1.0 created (see Appendix A). Retested as ID 9.1* |
| 1.1 | ConfigFile Function:  get\_soc() | Path for soc data in flt32 file format | Result = valid np array |  | *Jane Doe, 2-15-2019* | *Pass* | *Complete* |
| 1.2 | ConfigFile Function:  test\_write() | None | Result = valid output config file |  |  |  |  |
| 2.0 | Meteorological Input Function:  \_subset\_data(data\_list) | Data\_list: [‘MET’,’vpd’] | Result = returns valid array of numbers from data |  |  |  |  |
| 2.1 | Meteorological Input Function:  \_find\_pft\_to\_claimed\_sites() | Valid variable: \_pft\_grids | Result = array that calculates dominant pft for each site |  |  |  |  |
| 2.2 | Meteorological Input Function:  sites\_claimed\_by\_pft(pft) | pft: 0 | Result = array of tower sites that have dominant pft 0 |  |  |  |  |
| 2.3 | Meteorological Input Function:  compute\_climatological\_year(start\_date,end\_date,meteor\_variables) | Start\_date: datetime (2000, 1, 1), end\_date: datetime(2000, 12, 31),  meteor\_variables: [VPD, SMRZ, SMSF, TMIN, TSURF, TSOIL, FPAR, PAR] | Result = updates variable \_meteor\_vars with a 365-day climatological year |  |  |  |  |
| 2.4 | Meteorological Input Function:  lat\_long() | None | Result = returns valid array of [long,lats] |  |  |  |  |
| 3.0 | FluxTowerData Function:  set\_coords(coord\_array) | coord\_array: valid array of [long,lats] | Result = calls \_set\_weights() |  |  |  |  |
| 3.1 | FluxTowerData Function:  \_set\_weights() | Empty weights array | Result = array of weights (nonzero) |  |  |  |  |
| 3.2 | FluxTowerData Function: smooth\_outliers(met) | met: “gust” | Result = window size(an int) for gpp and reco smoothing functions (3.3 and 3.4) |  |  |  |  |
| 3.3 | FluxTowerData Function: smooth\_gpp\_outliers(met,window) | met: “gust”, window: 10,  Valid towers array | Result =  Calls SingleFluxTower gpp smooth |  |  |  |  |
| 3.4 | FluxTowerData Function:  smooth\_reco\_outliers() | met: “gust”, window: 10,  Valid towers array | Result =  Calls SingleFluxTower reco smooth |  |  |  |  |
| 4.0 | SingleFluxTower Function:  test\_read() | Relative csv path for SingleFluxTower | Result = prints single tower’s data |  |  |  |  |
| 4.1 | SingleFluxTower Function:  smooth\_outliers\_single\_var(var,met,window) | var: valid RECOs,  met: “gust”, window: 10 | Result = outliers from var are filtered |  |  |  |  |
| 5.0 | NewBPLUT Function:  test\_read() | Relative csv path for BPLUT | Result = prints BPLUT values |  |  |  |  |
| 5.1 | NewBPLUT Function:  gpp\_params(pft) | pft: 0 | Result = gets valid BPLUT GPP params for pft 0 |  |  |  |  |
| 5.2 | NewBPLUT Function:  reco\_params(pft) | pft: 0 | Result = gets valid BPLUT RECO params for pft 0 |  |  |  |  |
| 5.3 | NewBPLUT Function:  kmult\_params(pft) | pft: 0 | Result = gets valid TSOIL, SMtop min and max for pft 0 |  |  |  |  |
| 5.4 | NewBPLUT Function:  after\_optimization(gpp\_or\_reco, pft, vars\_optimized) | gpp\_or\_reco: “GPP”, pft: 0, vars\_optimized: all 8 parameters (new vals) | Result = updated BPLUT shows optimized values for pft 0 |  |  |  |  |
| 5.5 | NewBPLUT Function:  test\_write() | None | Result = updated bplut outputted in csv format |  |  |  |  |
| 6.0 | ReferenceInput Function:  subset\_by\_pft(pft,tower\_sites\_claimed\_by\_pft) | pft: 0, tower\_sites\_claimed\_by\_pft: valid array of tower sites | Result = subsets gpp and reco for these valid tower sites |  |  |  |  |
| 6.1 | ReferenceInput Function:  gpp() | None | Result = valid array of observed gpp form towers |  |  |  |  |
| 6.2 | ReferenceInput Function:  reco() | None | Result = valid array of observed reco from towers |  |  |  |  |
| 6.3 | ReferenceInput Function:  \_subset\_data(data\_list) | data\_list: [‘GPP’, ’gpp\_pft1\_mean’] | Result = valid arrays for observed gpp |  |  |  |  |
| 7.0 | PFTSelector Function:  select\_pft(meteor\_input,prev\_pfts) | meteor\_input: from previous test, prev\_pfts: [1,2] | Result =  Displays valid pfts 1-8 and displays previous pfts next to this |  |  |  |  |
| 8.0 | GPP Function:  clean\_nans(array) | array: [1.0,8.7,nan,9.8,5.76,nan,nan,19.8] | Result = array with no nans |  |  |  |  |
| 8.1 | GPP Function:  set\_apar\_bounds() | FPAR: array of floats,  PAR: array of floats | Result = ability to set bounds (int) |  |  |  |  |
| 8.2 | GPP Function:  func\_to\_optimize(gpp\_param) | gpp\_param:  bplut.gpp\_params(pft) where pft:0 | Result = returns valid SSE (11.0) |  |  |  |  |
| 8.3 | GPP Function:  gpp\_v\_emult(pft,bplut,gpp\_param) | pft: 0, bplut: valid BPLUT, gpp\_param:  bplut.gpp\_params(pft) | Result = displays prompt if wanting to view optional graph |  |  |  |  |
| 9.0 | Ramp\_Func Function:  downward\_ramp\_func(x,x\_min\_max) | x: 10.0, x\_min\_max:  (5.0,20.0) | Result = 0.5 |  |  |  |  |
| 9.1 | Ramp\_Func Function:  upward\_ramp\_func(x,x\_min\_max) | x: 10.0, x\_min\_max: (5.0,30.0) | Result = 0.25 |  |  |  |  |
| 9.2 | Ramp\_Func Function:  kmult\_arrhenius\_curve(x,bt\_soil) | x: 10.0,  bt\_soil: 5.00 | Result = 1.1038 |  |  |  |  |
| 9.3 | Ramp\_Func Function:  arrhenius\_curve(x,mult,a,b) | x: 10.0,  mult: 5.00 ,  a: 25.0,  b: 100.0 | Result = 1.2911 |  |  |  |  |
| 10.0 | GPP\_Func Function:  gpp\_apar(apar, vpd, tmin, smrz, tsurf, lue, vpd\_min, vpd\_max, tmin\_min, tmin\_max, smrz\_min, smrz\_max, ft\_mult\_frozen, ft\_mult\_thawed) | Apar: 10.0, vpd: 12.5, tmin: 15.0, smrz: 17.5, tsurf: 280.0, lue: 1.5, vpd\_min: 10.0, vpd\_max: 20.0, tmin\_min: 12.5, tmin\_max: 22.5, smrz\_min: 15.0, smrz\_max: 25.0, ft\_mult\_frozen: 0.5, ft\_mult\_thawed: 1 | Result = 1.07811 |  |  |  |  |
| 10.1 | GPP\_Func Function:  gpp(fpar, par, vpd, tmin, smrz, tsurf, lue, vpd\_min, vpd\_max, tmin\_min, tmin\_max, smrz\_min, smrz\_max, ft\_mult\_frozen, ft\_mult\_thawed) | fpar: 5.0,  par: 38.0, vpd: 12.5, tmin: 15.0, smrz: 17.5, tsurf: 230.0, lue: 1.5, vpd\_min: 10.0, vpd\_max: 20.0, tmin\_min: 12.5, tmin\_max: 22.5, smrz\_min: 15.0, smrz\_max: 25.0, ft\_mult\_frozen: 0.5, ft\_mult\_thawed: 1 | Result = 10.242045 |  |  |  |  |
| 10.2 | GPP\_Func Function:  emult(vpd, tmin, smrz, tsurf, lue, vpd\_min, vpd\_max, tmin\_min, tmin\_max, smrz\_min, smrz\_max, ft\_mult\_frozen, ft\_mult\_thawed) | vpd: 12.5, tmin: 15.0, smrz: 17.5, tsurf: 230.0, lue: 1.5, vpd\_min: 10.0, vpd\_max: 20.0, tmin\_min: 12.5, tmin\_max: 22.5, smrz\_min: 15.0, smrz\_max: 25.0, ft\_mult\_frozen: 0.5, ft\_mult\_thawed: 1 | Result = 0.035937 |  |  |  |  |
| 11.0 | SSE Function:  sse (all\_obs, all\_pred, non\_missing\_obs, weights) | all\_obs: flux\_tower\_data.gpp(), all\_pred: simulated\_gpp(), non\_missing\_obs: flux\_tower\_data.non\_missing\_observations(), weights: flux\_tower\_data.weights() | Result = valid list of gpp values |  |  |  |  |
| 11.1 | SSE Function:  rmse(n\_s, obs, pred) | n\_s: 2,  obs: [5.25,6.32], pred: [5.34,6.54] | Result = 4.20703 |  |  |  |  |
| 12.0 | RECO Function:  set\_prh\_and\_pk() | Specify prh as .75 and pk as .5 | Result = no error message to display for re-prompting |  |  |  |  |
| 12.1 | RECO Function:  \_simulate\_reco(reco\_param) | reco\_param: bplut.reco\_params(pft) for pft 0 | Result = valid array of simulated RECO values |  |  |  |  |
| 12.2 | RECO Function:  rhc\_v\_kmult() | None, input y | Result = displays optional graph |  |  |  |  |
| 12.3 | RECO Function:  func\_to\_optimize(reco\_param) | reco\_param:  bplut.reco\_params(pft) where pft:0 | Result =  returns valid SSE (11.0) |  |  |  |  |
| 13.0 | RECO\_Func Function:  kmult(t\_soil, smsf, bt\_soil, smsf\_min, smsf\_max) | t\_soil: 10.0, smsf: 15.0, bt\_soil: 5.0, smsf\_min: 12.5, smsf\_max: 22.5 | Result = 0.36793 |  |  |  |  |
| 13.1 | RECO\_Func Function:  reco(gpp, t\_soil, smsf, c\_bar, f\_aut, bt\_soil, a, b, smsf\_min, smsf\_max) | Gpp: 25.0, t\_soil: 10.0 , smsf: 15.0 , c\_bar: 12.5, f\_aut: 0.5 , bt\_soil: 5.0 , a: 25.0, b: 100.0, smsf\_min: 12.5, smsf\_max: 22.5 | Result = 16.535 |  |  |  |  |
| 13.2 | RECO\_Func Function:  reco(gpp, kmult, c\_bar, f\_aut) | Gpp: 6.08, kmult: 1.35,  c\_bar: 0.57,  f\_aut: 0.78 | Result = 6.40224 |  |  |  |  |
| 14.0 | AnalyticalModelSpinUp Function:  calc\_npp(towers\_gpp,fraut) | towers\_gpp: [[5.5,3.45],[2.34,7.68]], fraut: .78 | Result = [ [1.21,0.759],[0.5148,1.6896] ] |  |  |  |  |
| 14.1 | AnalyticalModelSpinUp Function:  summed\_kmults() | None | Result = valid array values |  |  |  |  |
| 14.2 | AnalyticalModelSpinUp Function:  get\_npps() | None | Result = valid array of array of npp values |  |  |  |  |
| 15.0 | SOC Function:  calc\_sigmas() | Num\_towers: 2, kmult: [1.5,3.0], npp: [2.5,4.0] | Result = [0.72,0.72] |  |  |  |  |
| 15.1 | SOC Function:  calc\_beta\_soc(fmet,fstr,kstr,krec,ropt) | Fmet: 0.456, fstr: 0.67, kstr: 0.34, krec: 0.5, ropt: 0.98 | Result = 0.002842 |  |  |  |  |
| 15.2 | SOC Function:  calc\_estimate() | sigmas = [0.72,0.54]  beta\_soc = .002842 | Result = [.00204624, .00153468] |  |  |  |  |
| 16.0 | NumericalModelSpinUp Function:  set\_iterations() | None, input an integer of 10 | Result = calling of \_forward\_run() 10 times |  |  |  |  |
| 16.1 | NumericalModelSpinUp Function:  \_forward\_run() | None, set in NumericalModelSpinUp creation | Result = calculation of pools |  |  |  |  |
|  | *…Add all rows necessary for a complete plan of test cases.* |  |  |  |  |  |  |

Table 2: Requirements Test Cases

|  |
| --- |
| Requirements Test Cases |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID\*** | **Test Case Summary** | **Prerequisites** | **Test Procedure** | **Input Data** | **Expected Result** |
| 1.0 | Calculate the dominant PFT for each tower site |  |  |  |  |
| 1.1  (MP01) | Calculate percentage of area that is occupied for each 9km tower site |  |  |  |  |
| 2.0 | Read in the input datasets from the configuration file for calibration |  |  |  |  |
| 2.1 (MP03) | Subset time series variables to configurable period |  |  |  |  |
| 3.0 | Gather L4C Meteorological inputs and ground-truth/observed data from the flux towers |  |  |  |  |
| 4.0 | User chooses one of the PFTs |  |  |  |  |
| 4.1  (MP04) | Subset tower sites to the chosen PFT |  |  |  |  |
| 4.2  (MP06) | Compile important data for optimization of GPP and RECO (ex: observed GPP, VPD, TMIN, etc) |  |  |  |  |
| 4.3  (MP05) | Compile ancillary info on each site for selected PFT (ex: SOC stock size for the site) |  |  |  |  |
| 5.0 | Guide user through removing outliers in average annual GPP and RECO calculations |  |  |  |  |
| 5.1  (LP01) | Remove negative values in annual GPP and RECO for each flux tower site |  |  |  |  |
| 5.1  (MP07) | Average data for PFT for each day of the year |  |  |  |  |
| 5.2  (MP19) | Calculate GPP\*, Kmult\*, NPP\* based off of optimized calculations |  |  |  |  |
| 6.0 | Allow user to optimize GPP and RECO parameters (8 GPP params, 4 RECO params) |  |  |  |  |
| 6.1  (MP11) | Allow user to specify Pk and Prh for RECO |  |  |  |  |
| 7.0 | Calculate linear ramp functions given current BPLUT |  |  |  |  |
| 7.1  (MP08,  MP12) | Calculate linear ramp functions using the updated values after optimization (and be able to save) |  |  |  |  |
| 7.2  (LP02) | Calculate lower and upper limits for APAR (user can set bounds) |  |  |  |  |
| 7.3  (MP09,  LP04) | Allow user to plot GPP against Emult and Rh/Cbar against Kmult as an optional graph |  |  |  |  |
| 7.4  (MP10) | Report differences between original and updated/optimized values |  |  |  |  |
| 7.5  (MP13) | Calculate Cbar for each tower site after optimization |  |  |  |  |
| 7.6  (LP03) | Allow the user to repeat optimizations after the initial optimization |  |  |  |  |
| 8.0 | Allow the user to specify the number of Numerical Spin-Up iterations |  |  |  |  |
| 8.1  (MP15) | Run Analytical and Numerical Model Spin-Ups |  |  |  |  |
| 8.2  (MP16) | Run preliminary spin up arbitrary period over full operational record 2000-2019 |  |  |  |  |
| 9.0 | Compute comprehensive validation (SOC) and fit statistics for each PFT |  |  |  |  |
| 9.1  (MP14) | Display Estimated SOC vs Ground-Truth SOC |  |  |  |  |
| 9.2  (MP20) | Calculate RMSE for each subset of towers for GPP and RECO |  |  |  |  |
| 10.0 | Output updated BPLUT, 4 SOC pool calculations, and report SMRZ min/max for years 2000-2018 (total forward model run) |  |  |  |  |
| 10.1  (MP21) | Calculate NEE (NEE= RECO – GPP) based on optimized parameters then report |  |  |  |  |
| 11.0 | Calculate flux tower weights as some tower sites are located within the same 9km area |  |  |  |  |
| ***Test Case IDs link back to SRS*** | High Priority Requirements are the #.0, not labeled with HP  Dependencies on the HP Requirements:  MP = Medium Priority Requirements  LP = Low Priority Requirements |  |  |  |  |

## Test Results (this section to be completed by Week 9)

The results of unit testing are added to the Unit Test Cases table in Section 2.5 (above), designated with the columns Actual Output, Tester Name & Date, Pass/Fail, and Resolution.

The results of functional testing are based off of the Requirements Test Cases in Section 2.5 (above), but having only the Summary, Input Data, and Expected Result. This table gives the results of these functional requirement test cases.

Table 3: Functional Testing Results

|  |
| --- |
| Functional Test Results |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID\*** | **Test Case Summary** | **Input Data** | **Expected Result** | **Actual Result** | **Tester name & date** | **Pass/Fail** | **Resolution** |
| 1.0 | *Login with a user incorrect account.* | *Username: abc*  *Password: abc* | *An error will be displayed for the wrong credentials.* |  | *Jane Doe, 2-15-2019* |  |  |
| 2.2 | *Add Challenge on Admin page with overlapping dates* | *Title: Title*  *Description: Description*  *Start Date: 03/01/19*  *End Date:03/01/18* | *Error Message displayed that start date must be before end date* |  |  |  |  |
| 3.5 | *Create new user* | *first name: abc*  *last name: abc*  *username: abc*  *email: abc@gmail.com password: abc*  *password again: abc*  *rent or own: rent*  *home size: 1000*  *household size: 2*  *other energy: no* | *User is successfully logged in, added to database, has all functionality available* |  | *Jane Doe, 2-15-2019* |  |  |
|  | *…Add all rows necessary for a complete plan of test cases.* |  |  |  |  |  |  |

***\*Test Case IDs link back to SRS***

# Acceptance and Approval

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] |  |  |

# References

1. Hamill, Paul. “Unit Test Frameworks.” *O'Reilly Online Learning*, O'Reilly Media, Inc. [www.oreilly.com/library/view/unit-test-frameworks/0596006896/ch01.html](http://www.oreilly.com/library/view/unit-test-frameworks/0596006896/ch01.html)
2. “Functional Testing.” *Functional Testing - an Overview*, ScienceDirect. [www.sciencedirect.com/topics/computer-science/functional-testing](http://www.sciencedirect.com/topics/computer-science/functional-testing)

# Appendices

*[Include any extra documentation in the appendices that supplements the main document text.]*