Report i-Tree Hydro

Summer 2013

Design & Implementation of a New iHydro Software Architecture

Prepared by:

David DiMaggio
CASE Program Manager
ddimaggi@syr.edu

Shannon Conley
Syracuse University Computer Engineering Graduate Student/ CASE Co-Op Intern
shannonvconley18@gmail.com

Prepared for:

Laura Welch
CASE Deputy Director
2-212 Center for Science and Technology
Syracuse University
lgwelch@syr.edu

Table of Contents

Introduction

This report details the contributions that I have made to the ongoing development of the i-Tree Hydro model. As the current CASE co-op program intern assigned to this project, I have worked along side the current iTree Hydro ESF graduate student, Thomas Taggart, to tackle the following main objectives

1. Prepare Version 5 for release [primarily the work of former graduate student, Yang Yang, and CASE intern, Yu Chen].
2. Create a new iTree Hydro architecture known as Version 6
3. Implement/add routing calculations to Version 6

The report will provide a brief overview of each objective, discuss the progress of each objective, and project the completion date of each objective.

Objective 1: Prepare Version 5 for Release

Overview

Before creating the new architecture, it was necessary to ensure that the Version 5 code, scheduled for release Spring 2012, was working correctly and performing comparably to the previously released Version 4 code.

Tasks

1. Speed up the performance of Version 5 simulations

Issue Description: As the simulation period increased, it was found that the Version 5 was significantly slower that Version 4.
Work Completed: Introduced a high-resolution timer package into both Version 4 and Version 5 to identify what Version 5 calculations were responsible for the performance hit. Determined that many accessor functions throughout the code were returning entire vectors as opposed to the desired vector element. Went through the code and changed all such instance to return single value as opposed to the vector.
Status: Completed.

2. Eliminate file parsing errors

Issue Description: Noticed that the eof() function used to end a file parsing while-loop was leading to an incorrect number of input elements.
Work Completed: Eliminate the use of eof() and replaced with >> operator.
Status: Completed.

3. Replace Julian date calculations with a different algorithm

Issue Description:
Found that LAI series calculation values differed between Version 4 and Version 5, although they should have been identical given the same input files/input settings.

Work Completed:
Found that the discrepancy was due to an error in the Julian date conversion. To fix, coded in a different Julian date conversion algorithm.

Status:
Completed.

4. Prepare input data from the same watershed for different code versions

Issue Description: To identify where Version 4 and Version 5 differed and why, a data set from the same watershed during the same time was needed. [Input files for each have different formats and data.]
Work Completed: Tom and I used pre-processor code to prepare input for Harbor Brook water year Version 4 and Version 5. Document composed so the process could easily be repeated. Another document was composed that detailed where [code packages] Version 4 and Version 5 differed and why.
Status: Version 4.5 and Version 6 distributed still need certain data input files prepped for Harbor Brook.

5. Compare output to ensure minor changes didn't have unintended consequences

Issue Description: While making these modifications to Version 5, Yang Yang was also making minor changes to the code. A method was needed to ensure that these changes were not have unintended effects.
Work Completed: A small program was created to compare output files and identify at what line the change first occurred.
Status: Completed.

6. Evaporation calculation/division by zero error

Issue Description: Found that if the first time step begins with a temperature less that 32 degrees, currentSMaxTree and currentShortVegTree are not calculated/remain zero. As a result, tree evaporation and short vegetation evaporation are un-defined [-1#IND] for many of the time steps.
Work Completed: Added the appropriate sMax calculation into TreeInterSnow and ShortVegInterSnow.
Status: Completed.

Objective 2: Create a New i-Tree Hydro Architecture

Overview [1]

The previous versions of i-Tree were built upon the same architecture. However, the following needs/desires prompted an overhaul of the past design and the creation of entirely new architecture:

- **Flexible/Extensible**-the ability to run different simulation modes, including lumped, semi-distributed, and distributed, through the same program
- **Maintainable**
  - -easy to add/substitute/remove process or calculation
  - -easy to change the format/content of output files or input files
  - -easy to change the data provided/retained during a calculation
  - -easy to debug/step through
- **Mobile-**easily used by/customized for other iTree tools

- **Understandable**-thorough documentation, simple names, easy for beginning programmers to understand/add code

Under the guidance of Dr. Jim Fawcett, a new iTree Hydro design was created based on his Parser program's architecture. The architecture of Version 6 has been designed to adhere to the following

- Abstract interfaces, ICalc, IProcess, IConfigHydro, and IOuputWriter follow the **Open/Closed Principle** and are never modified. The interfaces serve as a 'protocol' that derived classes implement. When a derived class needs to be changed the core components of the architecture do not need to be re-configured/re-designed.[2]
- The hydroprime package that runs the simulation never needs to be changed through the use of the **Liskov Substitution Principle**. Every hydroprime function that operates on a base class IProcess or IOutputWriter is able to 'operate successfully when a derived class object is substituted for the base object'/'need no information about the derived object'.[2]

[1] Design vocabulary and definitions from Software Design .ppt Dr. Jim Fawcett, CSE 687, Spring 2009

[2]Design vocabulary and definitions from Chapter 12 Object Orientated Deign Principles .ppt Dr. Jim Fawcett, CSE 687, Spring 2007

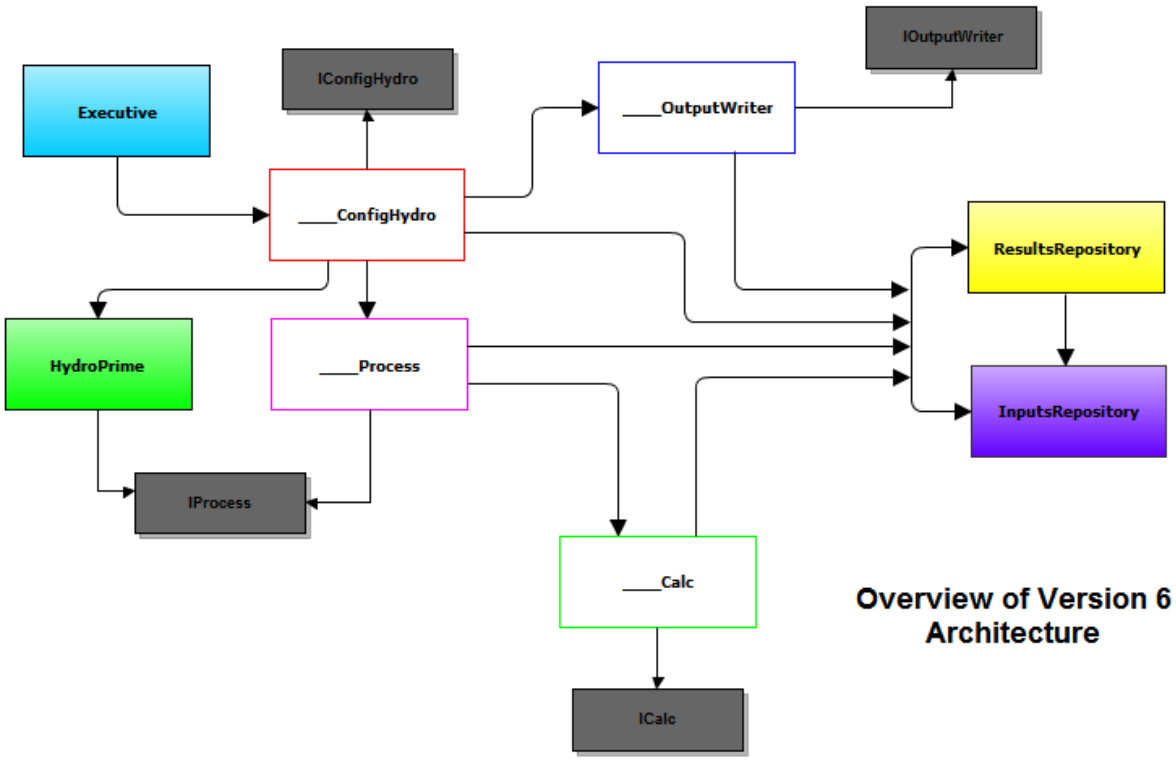
Tasks

1. Create design/prototype

Issue Description: Create a prototype/design to see if a design based on the architecture of Parser would be feasible.
Work Completed: Created the hydroprime prototype and then met with Dr. Fawcett, Yang Yang, Thomas Taggart, and Yu Chen to discuss it. It was decided that the prototype should be expanded to incorporate actual calculations/expanded and should be pursued.
Status: Completed.

2. Create core architecture

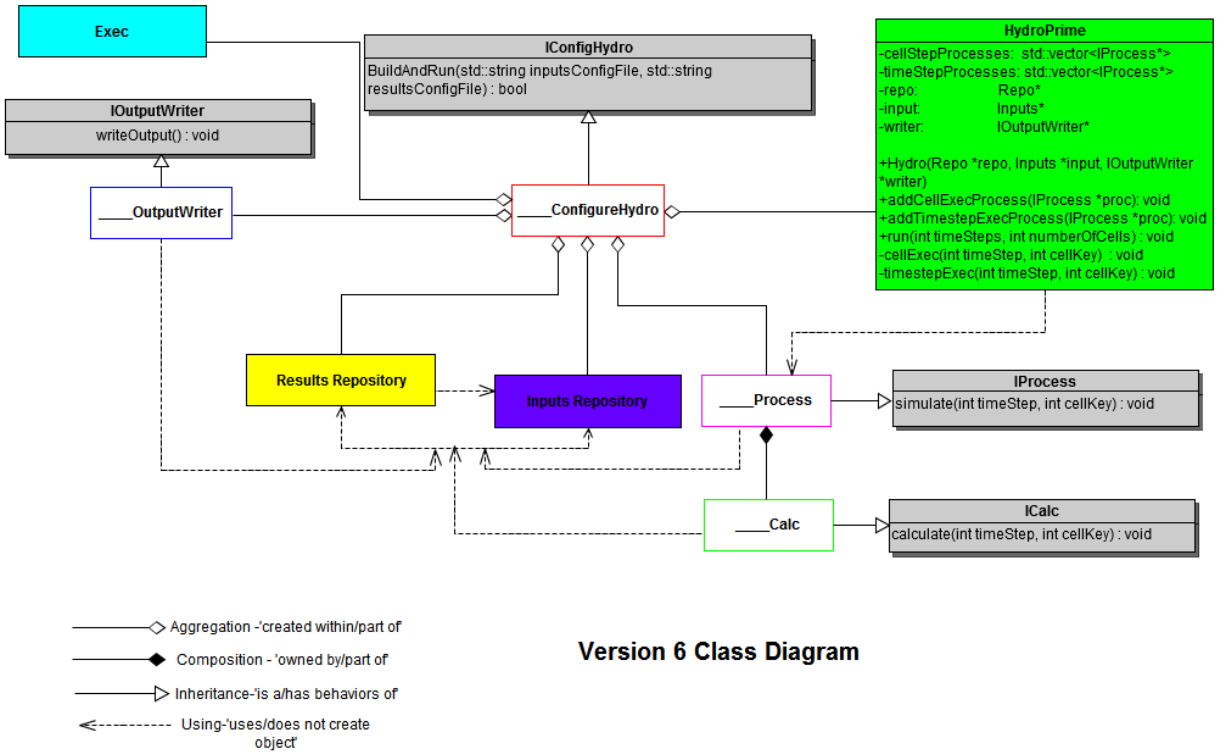
Issue Description: Create the core architecture needed to run a simulation.
Work Completed: The following diagrams have been included to illustrate the core architecture that was designed and implemented.



The diagram above provides a conceptual overview of the new iHydro design. The '__' fill-in-the blank symbol preceding the name of several titles indicates that the package (.h file or .h file and .cpp file) can be varied/tailored/interchanged for a specific simulation, but must follow/implement the interface it inherits. Interfaces in the diagram are colored grey and have names that begin with 'I'. On the other hand, the solid color packages are the same no matter what simulation mode is chosen/run. The packages shown in the figure serve the following purposes:

- **Executive** - Accepts user input and runs the desired simulation by calling the mode's ConfigHydro package
- **HydroPrime** - Runs the simulation by executing the processes specified in the ConfigHydro package
- **ResultsRepository** - Contains the data structures that store the results created by the calculation packages and used by the output writer packages during the creation of output files
- **InputsRepository** - Reads in the input files and stores information used by the calculation packages
- **ICalc** - Outlines the structure the calculation packages must implement to work within the architecture
- **__Calc** - A calculaton package that contains a hydrological calculation
  ex. **SnowMeltUnderShortVegCalc, TreeInterceptionCalc, SoilInfiltrationCalc, etc.**
- **IProcess** - Outlines the structure the process packages must implement to work within the architecture
- **__Process** - A process package that groups and calls the calculation packages that represent a hydrological process
  - ex. **InterceptionProcess, EvaporationProcess, ThroughFlowprocess**

    - **InterceptionProcess** package consists of the **TreeInterceptionCal**c, **ShortVegInterceptionCalc**
- **IConfigHydro** - Outlines the structure the configure hydro processes must implement to work within the architecture
- **__ConfigHydro** - For a given simulation mode, groups and sequences process packages, specifies the output writer package, and calls the hydroprime package to run the simulation
    - **ex. DistConfifHydro, SemiConfigHydro, V4_5DistConfigHydro** each have different process packages, different output writer packages, but all call the same hydroprime package
- **IOutputWriter** - Outlines the structure the output writer packages must implement to work within the architecture
- **__OutputWriter** - Creates the output files desired by a particular simulation mode
    - ex. **SemiOutputWriter**, **DistOutputWriter**



Version 6 Class Diagram

The class diagram provides a view of the relationships between Version6 classes. The interface classes [IConfigHydro, IOutputWriter, IProcess, ICalc] provide the function prototypes that must be implemented by the inheriting classes. The Executive class creates the configure hydro classes, while the configure hydro class creates the HydroPrime class, OutputWriter class, InputsRepository class, ResultsRepository class, and Process classes. Each process class creates a set of related calculation classes. OutputWriter, Process, and Calc classes are able to access/use the ResultsRepository and InputsRepository classes without having to create their own copies. At every time the step, hydroprime calls the simulate

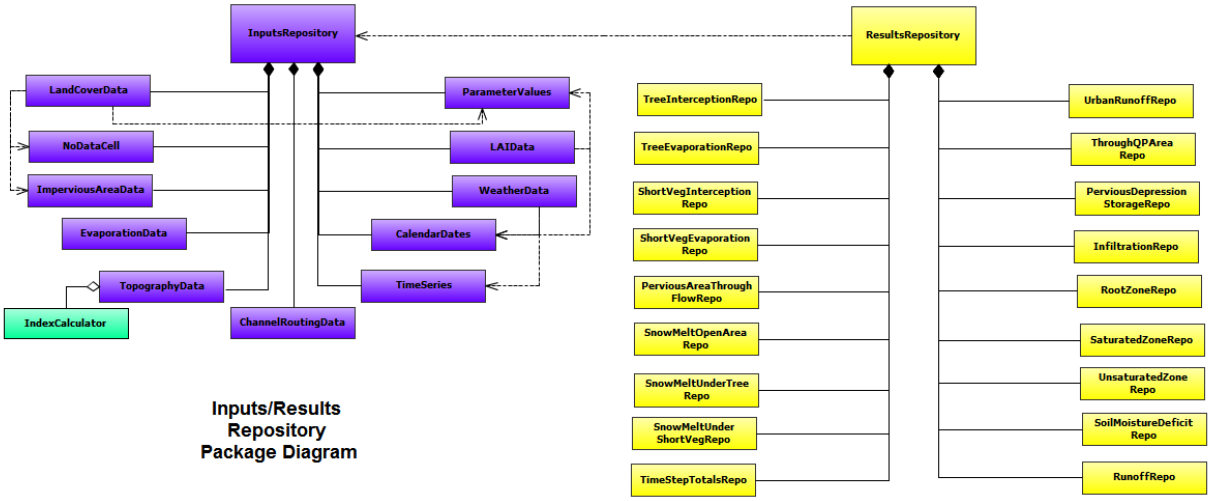
function for each process class in the timeStepProcesses container. For each cell, the simulate function for every process class in the cellStepProcesses container is called by hydroprime at every time step. The configure hydro class creates the processes and places them in either hydroprime container. Typically, a simulate function calls the calculate functions of the calc classes that belong/owned by that process class. For instance, the InterceptProcess simulate method calls the calculate functions that belonging to the TreeInterceptCalc and ShortVegInterceptCalc classes. The hydroprime class calls the simulate function via the IProcess pointer and does not have to be rewritten according to the process classes defined by a given configure hydro class.
Status: Completed.

3. Create method to ensure modes create output identical to corresponding versions

Issue Description: It was necessary to ensure accurate incorporation of the Version 4.5 and Version 5 calculations into the new architecture.
Work Completed: To do so, for each process, an output file or files was created that contained the results of a calculation in both the new version and the older version. The two files were compared to make sure that the values were identical for several watershed simulations.
Status: Completed

4. Create inputs and results repository data structures

Issue Description: In the results repository, structures are needed to store results created by calculation classes; however, data structures never belong to a calculation or process class even though names are often similar. In the inputs repository, structures are needed to read input data and store information used by calculation classes.
Work Completed: The following diagrams have been included to illustrate the core architecture that was designed and implemented.



Inputs/Results Repository Package Diagram

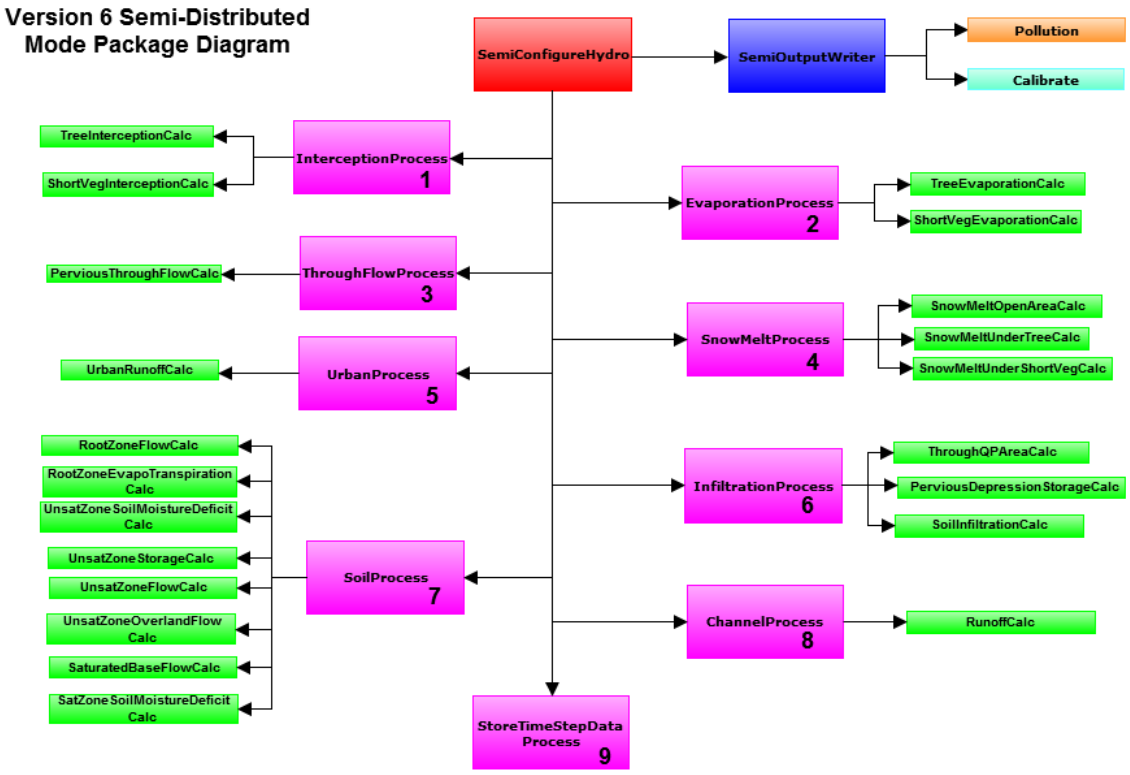
The figure above illustrates the organization of the InputsRepository and ResultsRepository. The structures that belong to the InputsRepository [purple] are responsible for reading, creating, and/or storing input data. The IndexCalculator class creates the Map.dat [Distributed mode] or Index.dat [Semi-Distributed mode] file via the Dem.dat file if the input directory does not contain already contain it. The ResultsRepository contains structures that store data related to a specific calculation. In some cases, repository data is initialized using InputsRepository values, which is why the ResultsRepository is given access to the InputsRepository object.

Status: Completed

5. Create semi-distributed hydrological simulation mode

Issue Description: A simulation model is needed to run a semi-distributed model where the calculations are based on Version 5.

Work Completed: The following diagrams have been included to illustrate the core architecture that was designed and implemented.



The diagram above shows the packages unique to the Version 6 mode that runs the semi-distributed hydrological simulation. The SemiConfigHydro package creates the process packages [purple] that contain a set of calc packages [green]. The numbers indicate the order in which each process's simulation is called by hydroprime for each simulation time step. The process and calculation order is listed below:

The following processes were added to HydroPrime's timeStepProcesses container:

- **InterceptionProcess**
    1. TreeInterceptionCalc
    2. ShortVegInterceptionCalc
- **EvaporationProcess**
    3. TreeEvaporationCalc
    4. ShortVegEvaporationCalc
- **ThroughFlowProcess**
    5. PerviousThroughFlowCalc
- **SnowMeltProcess**
    6. SnowMeltOpenAreaCalc
    7. SnowMeltUnderTreeCalc
    8. SnowMeltUnderShortVegCalc
- **UrbanProcess**
    9. UrbanRunoffCalc
- **InfiltrationProcess**
    10. ThroughQPAreaCalc
    11. PerviousDepressionStorageCalc
    12. SoilInfiltrationCalc
- **SoilProcess**
    13. V4_5RootZoneFlowCalc
    14. RootZoneEvapoTranspirationCalc
    15. UnsatZoneSoilMoistureDeficitCalc
    16. UnsatZoneStorageCalc
    17. UnsatZoneFlowCalc
    18. UnsatZoneOverlandFlowCalc
    19. TimeStepTotalCalc
    20. SaturatedBaseFlowCalc
    21. SatZoneSoilMoistureDeficitCalc
- **ChannelProcess**
    22. RunoffCalc
- **StoreTimeStepDataProcess**

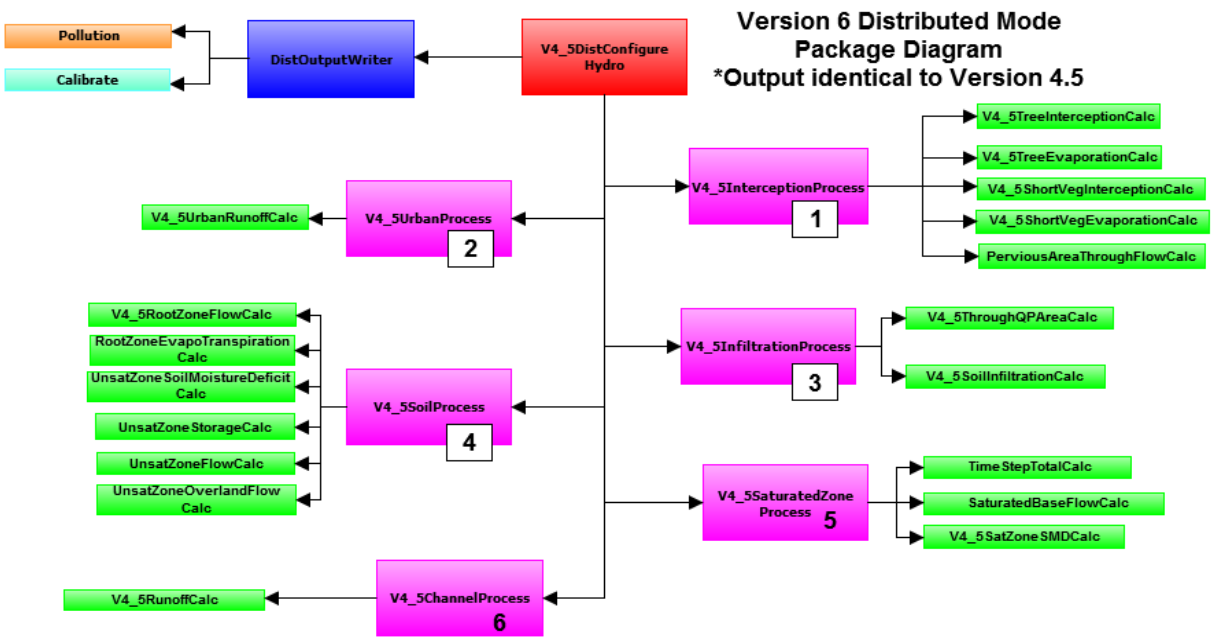
Status:
Almost Complete. The simulation still needs to be tested on more watersheds. Tom is also making minor changes to the calculations, which vary from Version 5

6. Create lumped hydrological simulation mode

Issue Description: Create a lumped hydrological simulation mode similar to either the semi-distributed or distributed mode except that the output values are medians and there is only a single topographic index.
Work Completed: Created lumped hydrological simulation model
Status: Completed

7. Create distributed hydrological simulation mode

Issue Description: A simulation model is needed to run a distributed simulation model. Version 4.5 has been as a basis for this mode. However, its calculations are based on Version 4 instead of Version 5. It was necessary to combine the logic of Version 4.5 and the Version 5 processes such as snow melt. To ensure that the distributed mode process was being incorporated accurately, a mode was first developed based solely on Version 4.5. Then, the more up-to-date Version 6 distributed mode was created.
Work Completed: The following diagrams have been included to illustrate the core architecture that was designed and implemented.



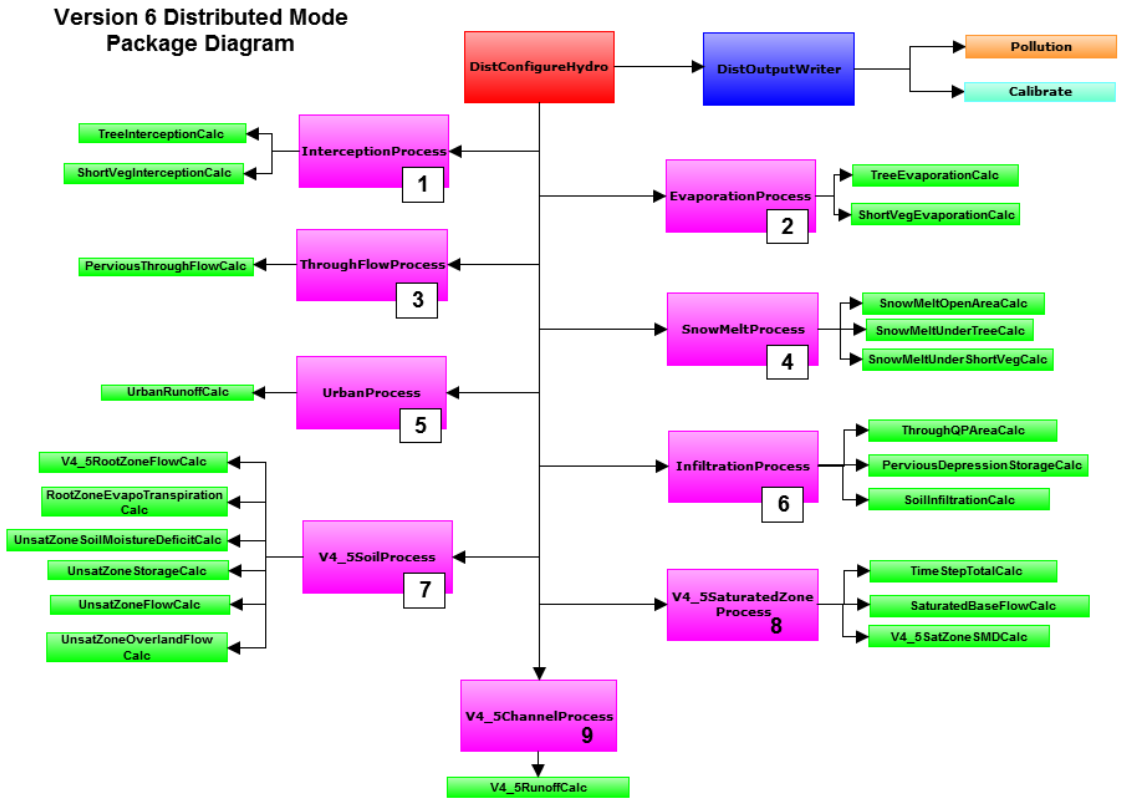
The diagram above shows the packages unique to the Version 6 mode that runs the distributed hydrological simulation that produces results identical to Version 4.5. The DistConfigHydro package creates the process packages [purple] that contain a set of calc packages [green]. The numbers indicate the order in which each process's simulation is called by hydroprime for each simulation time step. Numbers within a white box indicate that the process's simulation method is called for each cell at a given time step. The simulation and calculation order is listed below:

The following processes were added to HydroPrime's cellStepProcesses container

- **V4_5InterceptionProcess**
    1. V4_5TreeInterceptionCalc
    2. V4_5TreeEvaporationCalc
    3. V4_5ShortVegInterceptionCalc
    4. V4_5ShortVegEvaporationCalc
    5. PerviousThroughFlowCalc
- **V4_5UrbanProces**
    6. V4_5UrbanRunoffCalc
- **InfiltrationProcess**
    7. V4_5ThroughQPAreaCalc
    8. V4_5SoilInfiltrationCalc
- **V4_5SoilProcess**
    9. V4_5RootZoneFlowCalc
    10. RootZoneEvapoTranspirationCalc
    11. UnsatZoneSoilMoistureDeficitCalc
    12. UnsatZoneStorageCalc
    13. UnsatZoneFlowCalc
    14. UnsatZoneOverlandFlowCalc

The following process added to HydroPrime's timeStepProcesses container:

- **V4_5SaturatedZoneProcess**
    15. TimeStepTotalCalc
    16. SaturatedBaseFlowCalc
    17. V4_5SatZoneSMDCalc
- **V4_5ChannelProcess**
    18. V4_5RunoffCalc



The diagram above shows the packages unique to the Version 6 mode that runs the distributed hydrological simulation. The DistConfigHydro package creates the process packages [purple] that contain a set of calc packages [green]. The numbers indicate the order in which each process's simulation is called by hydroprime for each simulation time step. Numbers within a white box indicate that the process's simulation method is called for each cell at a given time step. The simulation and calculation order is listed below:

The following process added to HydroPrime's cellStepProcesses container:

- **InterceptionProcess**
  1. TreeInterceptionCalc
  2. ShortVegInterceptionCalc
- **EvaporationProcess**
  3. TreeEvaporationCalc
  4. ShortVegEvaporationCalc
- **ThroughFlowProcess**
  5. PerviousThroughFlowCalc
- **SnowMeltProcess**
  6. SnowMeltOpenAreaCalc
  7. SnowMeltUnderTreeCalc
  8. SnowMeltUnderShortVegCalc
- **UrbanProces**
  9. UrbanRunoffCalc
- **InfiltrationProcess**
  10. ThroughQPAreaCalc
  11. PerviousDepressionStorageCalc
  12. SoilInfiltrationCalc
- **V4_5SoilProcess**
  13. V4_5RootZoneFlowCalc
  14. RootZoneEvapoTranspirationCalc
  15. UnsatZoneSoilMoistureDeficitCalc

16. UnsatZoneStorageCalc
17. UnsatZoneFlowCalc
18. UnsatZoneOverlandFlowCalc

The following process added to HydroPrime's timeStepProcesses container:

- **V4_5SaturatedZoneProcess**
  19. TimeStepTotalCalc
  20. SaturatedBaseFlowCalc
  21. V4_5SatZoneSMDCalc
- **V4_5ChannelProcess**
  22. V4_5RunoffCalc

Status: Completed

Future Work/Feedback

In theory, another program would more than likely adapt this program for its own needs by taking the following one or more of the following steps:

1. Create inputs structures based on the data input files it would to utilize
2. Create the repo structures to store calculation data
3. Create the calculation classes
4. Group calculation classes into process classes
5. Create an output writer class to write output file(s)
6. Create an new ConfigHydro class to build the simulation mode
7. Outside program will either create the ConfigureHydro object or add it as an option to the exec package and run the process from within the program

The following suggestions/incorporation strategies were proposed /discussed for iDesign/iEco:

- This program will run the executable and use the lumped simulation mode
- The input parameters chosen by users might be restricted/chosen/optimized in advance
- Will probably require the design of a new output writer package
- The project will start in the next two weeks

The following suggestions/incorporation strategies were proposed /discussed for for iLandscape:

- Whether or not iLandscape will use an executable or use a ConfigHydro library via CLI is still unclear
- iLandscape will utilize the distributed processes
- The project will not begin in the immediate future since the current focus is the upcoming iTree release

The following suggestions regarding the current architecture were made:

- Create a system to catch and output input file errors
- A version of the code that uses exponential decay in calculations will be created and tested
- The use of distributed input files for weather and soil values will be investigated

- Rather than create new watersheds to test semi-distributed simulation, vary the current watershed inputs to test behavior in extreme conditions
- A calculation to estimate the run time given the input values/parameter configuration will be created
- Replace Version 6 with Version 5 in the 3-D curve code
- Format distributed output files
- Examine current soil process distributed calculations for accuracy
- Examine map.dat file creation accuracy
- Add flow routing

- Ensure that all input/output stream objects successfully created. If not, create either an exit strategy [terminate program if insufficient input] or an alert strategy [write to console/or log file if output file can not be created]
- Reset pointers to zero after deletion
- Make sure vectors are not being accessed even if an error occurs where the vector size is less than it should be
- Create a TimeStepTestProcess where more data is stores than usual. This data will be written to output file(s) and compared to Version 5. Version 5 will also have to be modified to produce the same output.
- Make sure input/results data structures store data only for the simulation begin and end periods
- Create unit tests for each package that contains calculations/file parsing

Objective 3: Implement Routing Calculations

Overview

Tasks

1. Determine which flow routing algorithm to incorporate

Issue Description: There a various flow routing algorithms that exist. Some will more computationally exhaustive than others. Depending on which is chosen, it might be necessary to implement parallel programming technique.
Work Completed: Ported DHSVM, a UNIX program written in C that runs a distributed hydrological simulation, to windows and created a visual studio solution to step through this code. This model utilizes DEMON flow-routing algorithm. However, the model literature indicates that it also possible to run D8. Tested DEMON but also tested the MFD (multi-flow direction) and D-Infinity routing algorithms.
Status: Completed

2. Introduce a flow routing algorithm into the distributed mode simulation

Issue Description: The iTree team/Tom have decided to focus on implementing D8 first. However, the possibility of using another method has not been ruled out.
Work Completed: The iTree team/Tom has coded up MFD and D-Infinity to complement D8 and these routines have been tested and made more efficient. The coding worked from algorithms initially coded by Yang Yang.
Status: Completed.

Conclusion

- **Objective 1** – Version 5 iTree Hydro was unofficially released in August 2013 to beta testers.
- **Objective 2-** The new architecture has been built and the semi-distributed simulation mode is almost finished. Attention was given to complete the distributed simulation mode and create a lumped simulation mode. The software is ready to be incorporated into other iTree programs such as Eco and Design.
- **Objective 3** – The introduction of a routing flow calculation into the distributed mode simulation was completed to allow for static and perhaps dynamic semi-distributed simulation mode.