Prey-Predator Simulator
Application Engineering Environment
And Process

Target Audience:
Application Engineer(s)

CS 673 Software Design & Production Methodology
Professor Ali Mili

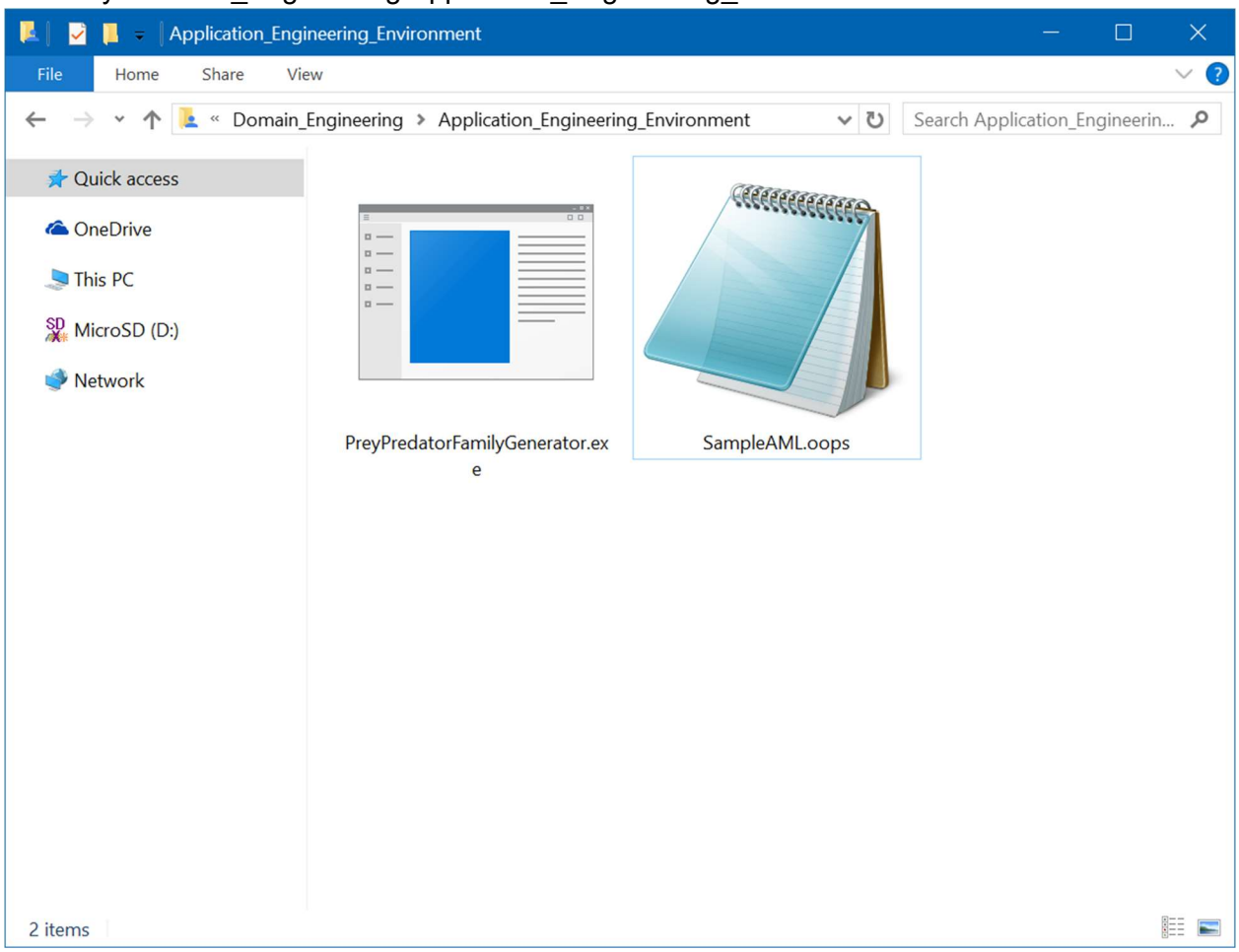
Team Putin
Rohit Lakshmana
Harsha Thotakura
Vladimir Ventura
Prashanth Reddy Motati

Table Of Contents

Application Engineering Environment

Location

Our prey-predator simulator application engineering environment is located in the following directory: Domain_Engineering\Application_Engineering_Environment. See screenshot below:



In this directory, you should find two files called “PreyPredatorFamilyGenerator.exe” and “SampleAML.oops”. However, the file “SampleAML.oops” should be replaced with the real OOPS (Object-Oriented Predation Specification) AML file for application engineering. (Note that executable will only run on a machine with the Windows OS).

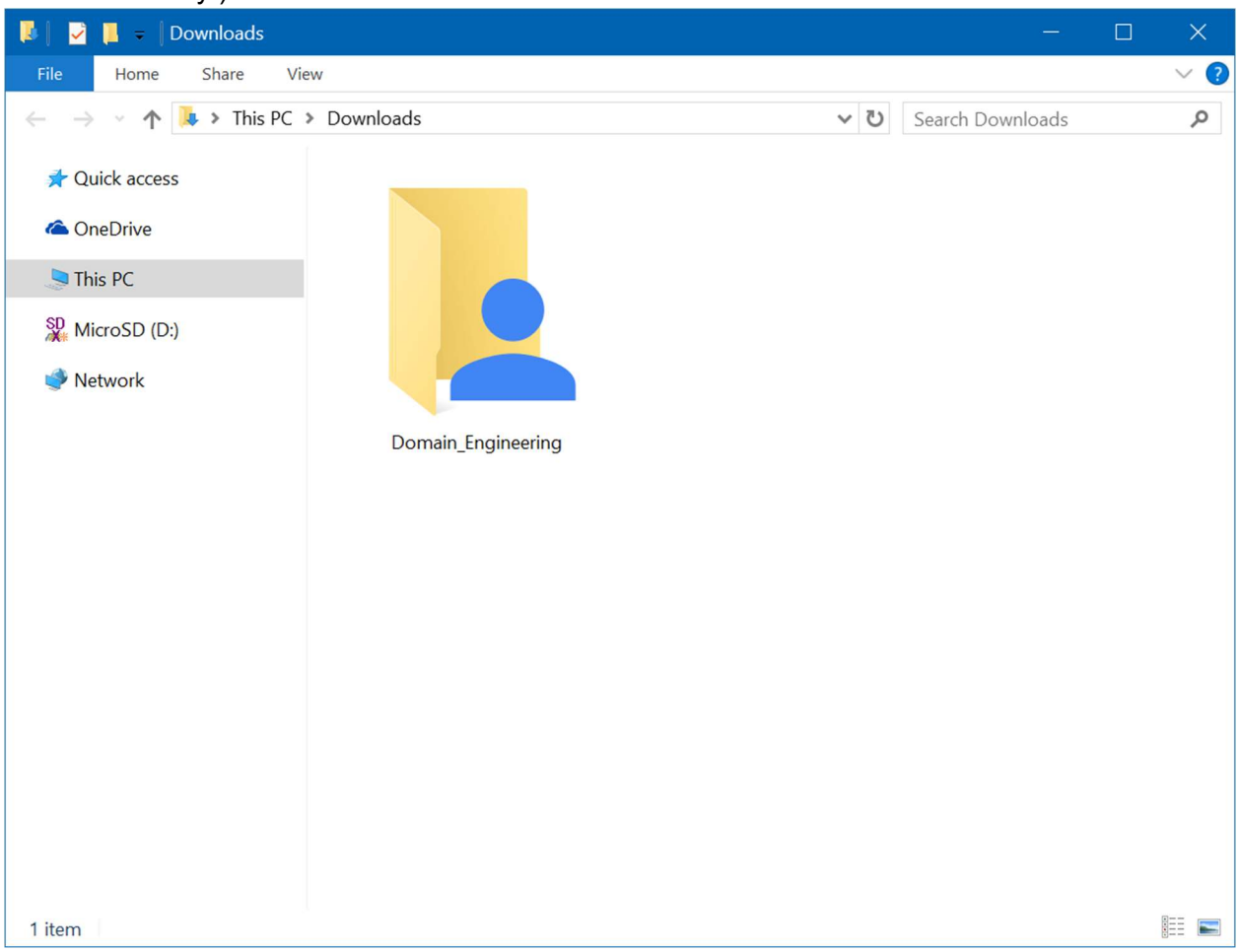
Execution

In order to run the executable “PreyPredatorFamilyGenerator.exe”, ensure that you have the .NET framework of 3.5 or more installed on your PC. Also, ensure that you have MATLAB

R2015a or R2015b installed so that you will be able to edit and run the MATLAB files generated by the “PreyPredatorFamilyGenerator.exe” executable.

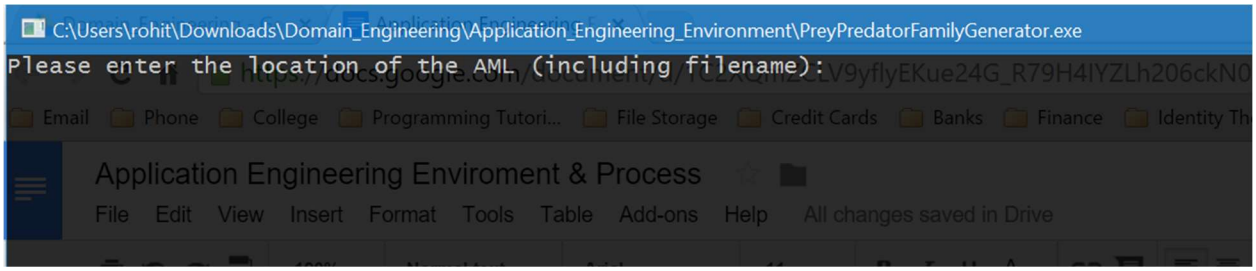
Below is a step-by-step explanation of how to use the application engineering environment.

1. Once you have the above environments setup, unzip and copy the entire “Domain_Engineering” folder to a location of your choosing. (For example see screenshot below, where we have copied it to the C:\Users\<username>\Downloads directory.)

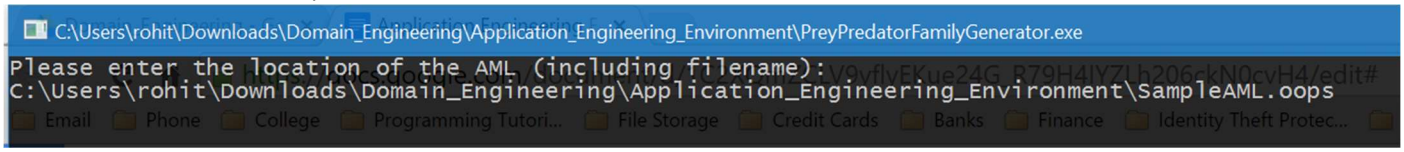


2. Head into the “...\Domain_Engineering\Application_Engineering_Environment” folder and double-click on the “PreyPredatorFamilyGenerator.exe” executable. You should be presented with the following screen with the dialog:

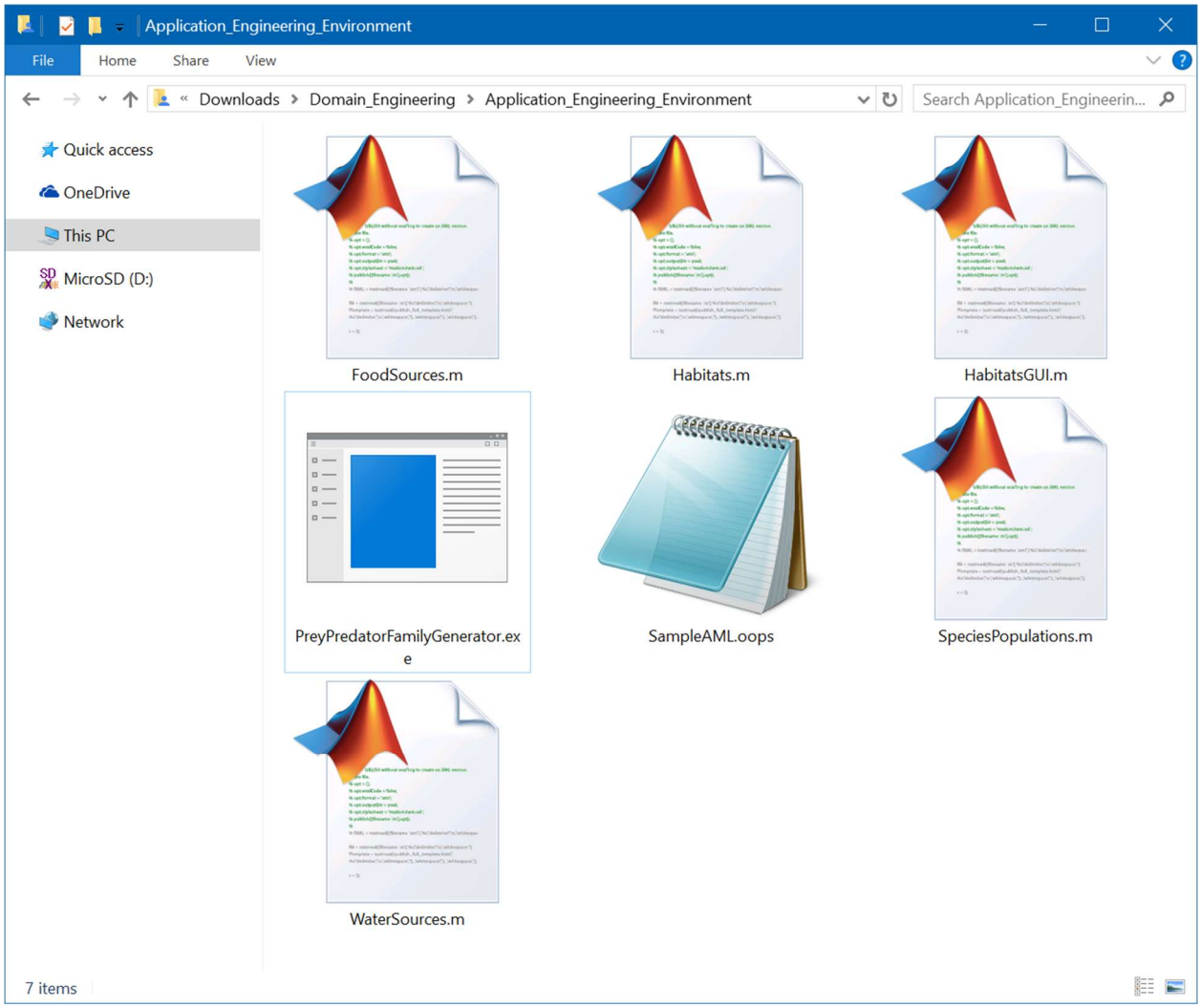
“Please enter the location of the AML (including filename):”



3. Enter the location of your OOPS AML (refer to the “Application Engineering Process” section below and the “OOPS Script Writer's Guide.pdf” document to learn how to create the OOPS AML).
   In this example, we will use the “SampleAML.oops” located in “C:\Users\<username>\Downloads\Domain_Engineering\Application_Engineering_Environment”. (Note: you must include the filename of your AML in the path, e.g C:\Users\rohit\Downloads\Domain_Engineering\Application_Engineering_Environment\SampleAML.oops)



4. Press the “Enter” key on your keyboard. The black command line dialog will disappear, however, you should see the following files generated the folder where you double-clicked the executable:



5. The files you see here auto-generated by the “PreyPredatorFamilyGenerator.exe” executable are all MATLAB template files for the prey-predator simulator. Note that individual applications of the prey-predator simulator are coded in MATLAB. The “HabitatsGUI.m” file is the front-end MATLAB GUI code that allows the user to input values other than what is specified in the OOPS AML. The rest of the “.m” files are the back-end algorithms that take in input values from the “HabitatsGUI.m” and calculate the species and food sources populations.
6. The “.m” files in the screenshot above are actual MATLAB code files that can be edited manually. If certain customer requirements in the application engineering phase require new features or modifications to existing features, the MATLAB files may be opened in MATLAB R2015a or R2015b, edited, and run. Refer to the “Application Engineering Process” below.
7. Currently, individual applications of the prey-predator simulator are not deployable because of the lack of the MATLAB Compiler SDK licences. MATLAB Compiler needs to be bought from MathWorks and installed to create stand-alone applications of the prey-predator simulator. As a work-around, you may continue to use the MATLAB development environment (IDE) to run, test, and debug applications.

Design

The design of the application engineering environment is fairly straight-forward. But it is important to re-visit the Commonalities and Variabilities as well as the Functional Capabilities described in Domain Analysis as this was what was used to create the Application Engineering environment:

Commonalities
The following are the common elements all PPS must have.

C1. In a simulation, there will always be a *set of species* in a habitat with an *initial size* for each specie.
C2. There will always be a certain amount of interaction between *species*, called the *interactivity coefficient*.
C3. Species will always die off at a certain rate due to predation, disease, or old age, called the *death rate*.
C4. Similarly, species will be born at a certain rate, called the *birth rate*.
C5. There will always be *vegetation growth rate* and *clean water growth rate*, which species depend on.

The following are elements that affect the *interactivity coefficient*, which every PPS must have.

CI1. Each specie will have an *aggressive factor* that describes its aggressivity towards preys, and an associated *aggressive factor multiplier* that describes if anything in the environment allows the specie to be more aggressive..
CI2. Each specie will have an *evasiveness factor* that describes its ability to evade predators, and an associated *evasive factor multiplier* that describes if anything in the environment allows the specie to be more aggressive..
CI3. Every PPS will have a scaling factor, based on the environment (area, terrain), which determines how much interactivity can occur.

Variabilities

The following are elements that PPS may vary on.

V1. The *habitat* will vary per simulation. In particular, the type of *habitat* will affect the availability and growth of water and vegetation, described by the *water and food multipliers*, and the ability for species to be more aggressive or evasive, described by the *aggressive and evasive multipliers*.
V2. The *season* will vary per simulation. In particular, Spring and Fall will generate more water; Summer and Winter will *generate less water*; and Winter will have a negative growth rate on water (as it freezes over).
V3. The *hunting style* will vary per simulation. In particular, the species can *hunt as a pack* or *hunt alone.*

V4. The *disease rate* will vary per PPS. The disease rate can become *an epidemic*, it can be *fully resisted*, or it can be anywhere in between for some species.

Parameters of Variation

Table P-1 shows the parameters of variation and their relationships to the variabilities.

Parameter	Meaning	Value Range	Binding Time	Default Value
V1. Habitat	Water, food, aggressiveness and evasiveness multipliers differ by habitat.	(FOREST, MEADOWS, DESERT, PRAIRIE, SWAMP, MOUNTAINS, ANTARCTIC) where each habitat has a different AFM, EFM, WM, and FM.	Specification	MEADOWS
V2. Season	Each season affects vegetation and water availability and growth rate (e.g. SPRING has higher water growth than WINTER)	(FALL, WINTER, SPRING, SUMMER) where each value affects growth rate for vegetation and water.	Specification	FALL
V3.HuntingStyle	The parameter is a scale of how well species hunt.	[0,1] where 1 indicates a species hunts in a pack; 0 indicates a species hunts alone. This directly affects aggressiveness factor multiplier *AFM*.	Specification	1
V4. Disease	The parameter is a scale of resistance towards disease	[0,1] where 0 indicates 0 resistance (epidemic will occur), and 1 indicates full	Specification	0

		resistance.		

Functional Requirements
Below lists the functional requirements the PPS family will be capable of performing to a user.

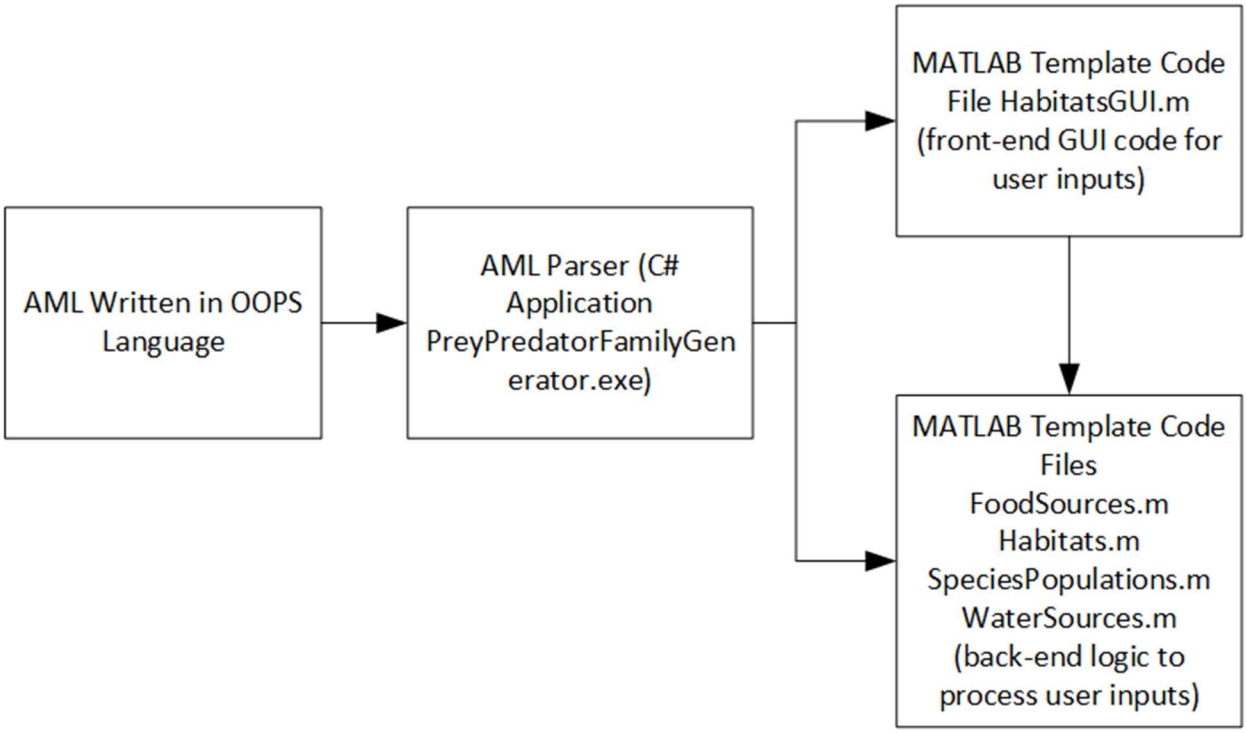
Functional Requirement ID	Functional Requirement
FR-1	The prey-predator simulations (PPS) software shall provide a front-end graphical user interface to allow for user-to-system interaction.
FR-2	The PPS software shall allow the user to provide the following user inputs: ● The habitat and its variabilities to conduct the simulation. ● Land in acres ● The set of all species in the habitat. ● Initial specie population for each specie ● Specie birth rate ● Specie death rate ● Specie aggressiveness factor ● Specie evasiveness ● Specie hunting rate & style ● Specie resistance to disease ● Initial water quantity
FR-3	The PPS software shall provide a graph to monitor the prey and predator populations on a week by week census.
FR-4	The PPS software shall provide a graph to monitor solid food sources and water quantity on a week by week census.
FR-5	The PPS software shall use the differential equations provided by the professor and in-class notes to display the results of the plots.

The application engineering environment developed satisfies all the commonalities and variabilities as well as the resulting functional capabilities.

The logic behind the design is as follows:

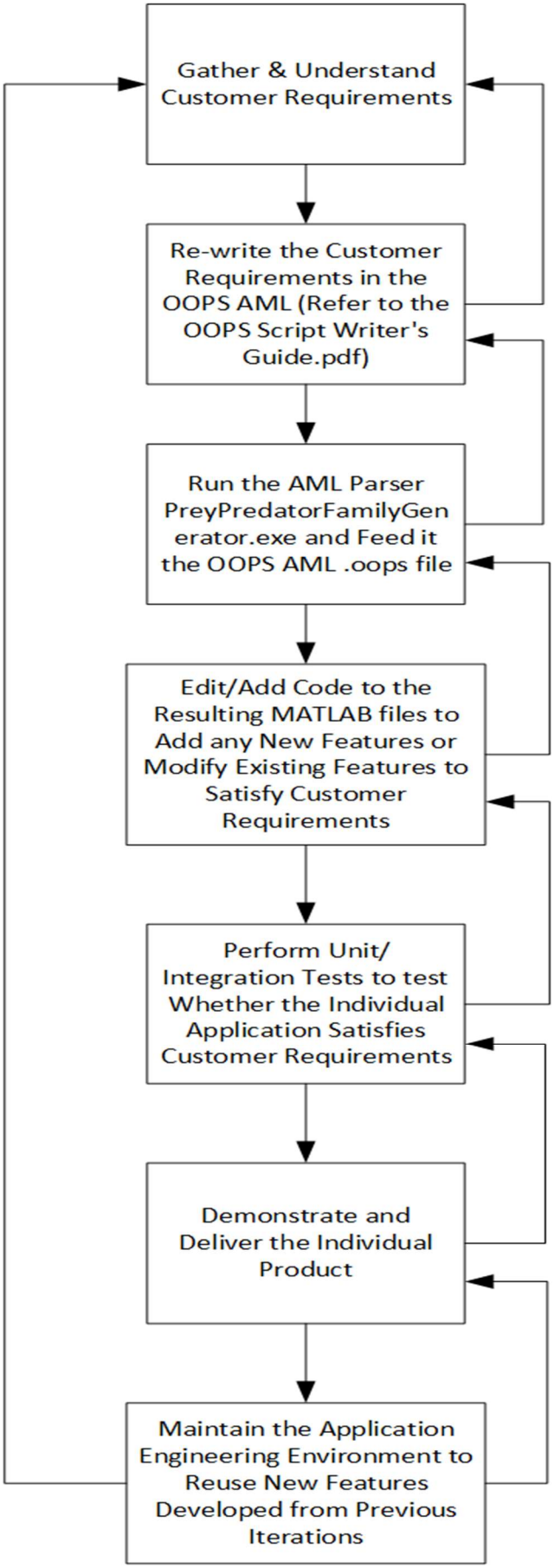
1. We created a test AML written in the OOPS AML language and included a few habitats with a few species and all the relevant information. (See the SampleAML.oops file).
2. We then created a C# application to build the “PreyPredatorFamilyGenerator.exe” AML compiler and software family member generator. The C# application was developed using the Visual Studio 2013 Ultimate IDE.

3. Within the C# application, logic was created & implemented to produce the dialog “Please enter the location of the AML (including filename):” and once the user entered the path to the AML, the C# application opened and read the AML file.
4. As the C# application reads the AML file, relevant information about the habitats, species, etc. are collected and structured into a C# collection object. This is known as our AML parsing phase.
5. Once the entire AML file is read and information stored in C# collection objects, the C# application uses the collection objects to auto-generate the MATLAB code files to build the code for the front-end GUI and back-end logic implementing differential equations mentioned in the Domain Analysis document. The parser (which is otherwise known as “PreyPredatorFamilyGenerator.exe”) literally hard-codes MATLAB keywords and logic into the relevant .m files based on what was specified in the OOPS AML.
6. The entire design can be summarized at a high-level via the following diagram:



Application Engineering Process

The Application Engineering Process is described by the following diagram:


Gather & Understand Customer Requirements
Re-write the Customer Requirements in the OOPS AML (Refer to the OOPS Script Writer's Guide.pdf)
Run the AML Parser PreyPredatorFamilyGenerator.exe and Feed it the OOPS AML .oops file
Edit/Add Code to the Resulting MATLAB files to Add any New Features or Modify Existing Features to Satisfy Customer Requirements
Perform Unit/ Integration Tests to test Whether the Individual Application Satisfies Customer Requirements
Demonstrate and Deliver the Individual Product
Maintain the Application Engineering Environment to Reuse New Features Developed from Previous Iterations

References

- MATLAB GUI Building Example: http://www.mathworks.com/help/matlab/creating_guis/about-the-simple-programmatic-gui-example.html
- Constructing GUI controls in MATLAB: http://www.mathworks.com/help/matlab/ref/uicontrol.html
- The full MATLAB documentation: http://www.mathworks.com/help/matlab/