Domain Analysis of

Family of

Prey-Predator Simulations



CS 673 Software Design & Production Methodology

Professor Ali Mili

Team Putin:

Rohit Lakshmana,

Harsha Thotakura,

Prashanth Reddy Motati,

Vladimir Ventura

## Table of Contents

|  |  |
| --- | --- |
| [Table of Contents](#h.fy7ckhmb8x2z) | 1 |
| [Introduction to Prey-Predator Simulations Product Line](#h.9hcgv75erfsk) | 2 |
| [Legacy Systems](#h.jcd22wiro0s1) | 2 |
| [Domain Qualification for Prey-Predator Simulations](#h.1he2ff6wn8vm) | 4 |
| [Dictionary of Terms and Equations](#h.v97a9hthr5sl) | 7 |
| [Scoping Commonality and Variability Analysis](#h.guq35yxlhkr8) | 10 |
| [Functional Requirements](#h.dn4b354bkvbk) | 12 |
| [Decision Model For Prey and Predator](#h.w48yt4ypvis8) | 13 |
| [Family Design Modules](#h.nbsfer6zhx95) | 13 |
| [FD1. Graphical User Interface Modules](#h.ez14vlo8935p) | 13 |
| [FD2. Software-Design-Hiding-Modules](#h.823ux2w3p63l) | 14 |
| [OOPS Application Modelling Language (AML)](#h.qyd3e214a15r) | 14 |
| [Uses Relations And Composition Mapping](#h.g5k4q2pwd1ab) | 19 |
| [References](#h.yybi9jbo26l7) | 21 |

## Introduction to Prey-Predator Simulations Product Line

Predation is a common phenomenon dictated by large populations between species in certain habitats. It is a density-dependent limiting factor on population growth of multiple species. Scientists and mathematicians have now developed multiple formulas and models that model and accurately predict population growth and predation between species. And very recently, with the emergence of computers and software technology, new software are being developed in order to simulate and graphically represent these predation models.

The Prey-Predator Simulation (PPS) product line gives biologists and interested parties a way to simulate and predict evolution of a set of species based on prey-predator relations under a variety of factors. These factors may include but are not limited to: the habitat, the number of species in a habitat, relations between species, sources and volumes of food, rates of predation (hunting and evasiveness factors), birth rates, death rates, climatic factors, and epidemics such as disease.

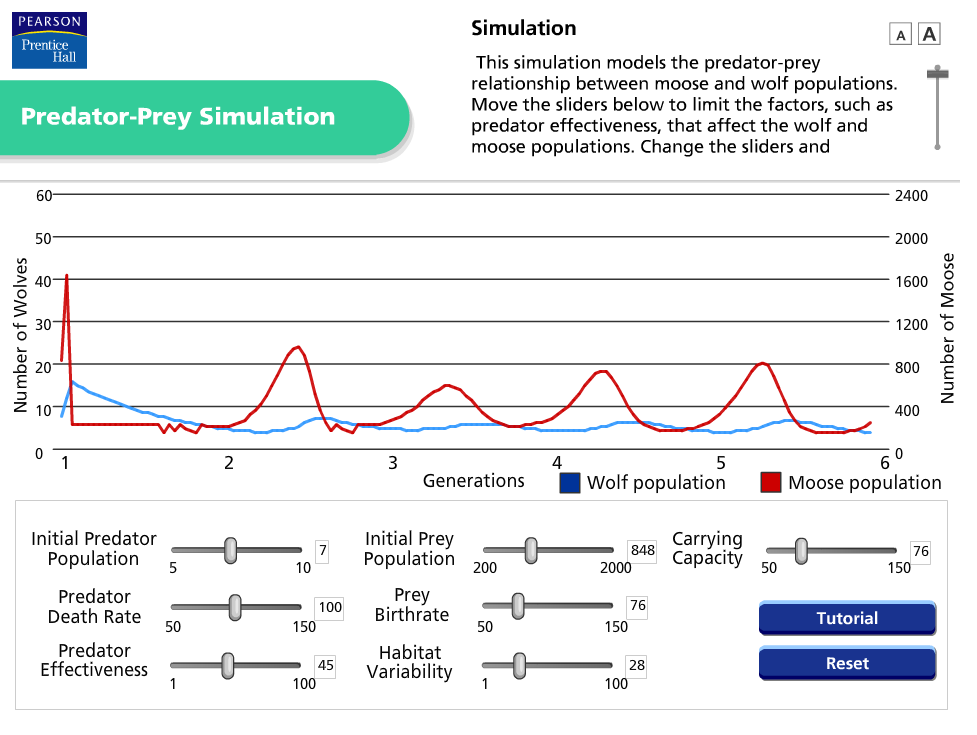
A typical application generated by an application engineer would provide a week by week census of each specie along with the volumes of each food source displayed as plots measured over time. A front-end GUI would provide those plots as well as controls to allow the user to conduct the simulation.

The scope of this document is to provide a complete domain analysis of a family of the PPS software (taking into account the above factors) identifying the results of domain scoping, domain assumptions and formulas, a decision model, a family design, and specifying an application modelling language to specify specific applications generated from the family. The domain analysis conducted here would then help application engineers to quickly generate specific applications within the prey-predator simulation product line according to customer requirements. The artifacts for the application engineering environment and process will be covered in a separate document.

## Legacy Systems

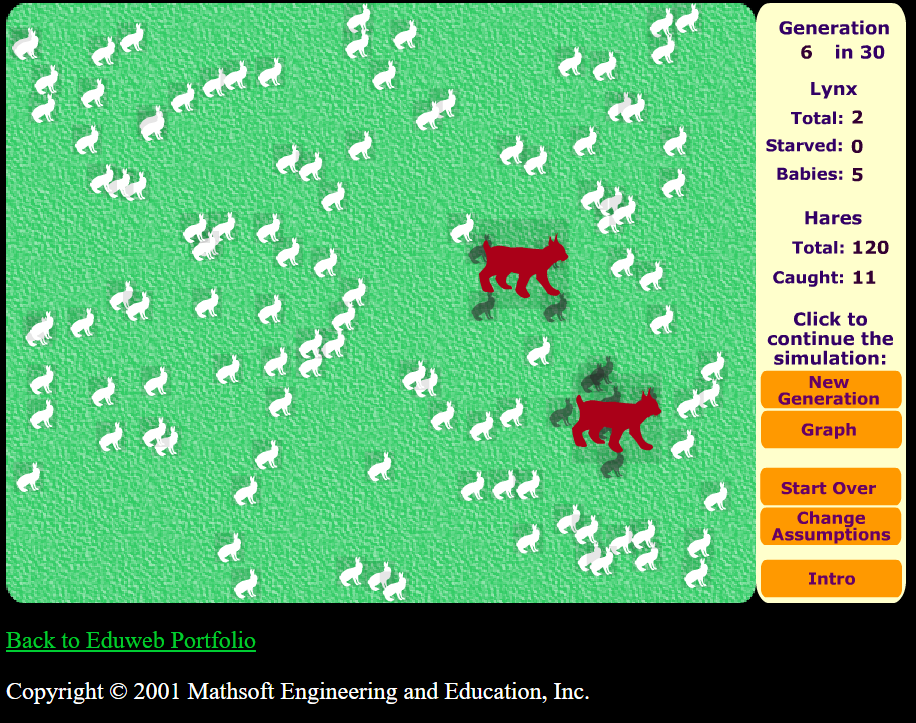
There are already several prey-predator simulation software in the market. One example can be found here: <http://www.phschool.com/atschool/phbio/active_art/predator_prey_simulation/index.html>

This specific application models the populations of moose (prey) and wolves (predator). The application provides controls to vary inputs such as: initial predator population, initial prey population, carrying capacity of the habitat, predator death rate, prey birthrate, predator effectiveness (which is simply an average of how successful the predator is in capturing the prey), and habitat variability (external factors that affect the populations). An example simulation of this application is captured in the below screenshot:



Another legacy system identified is a pictorial representation of the habitat: <http://www.eduweb.com/portfolio/studyworks/predators8a.html>

This simulation software focuses on lynx (predator) and hares (prey). The software allows the user to change some assumptions such as the number of hares a lynx must catch to survive and reproduce, maximum number of babies a lynx may have, maximum number of hares one lynx may catch, survival rate of hares when population reaches limit of food supply. An example run of the software application is captured below:



Of course the domain analysis conducted in this document will specify even more controls for other possible inputs and more graphs to display volumes of food sources. And more importantly, the analysis will focus on a family of new applications that could be generated rather than on specific applications.

## Domain Qualification for Prey-Predator Simulations

We start the domain qualification phase by asking few questions like – “Is the prey-predator simulations model economically viable? Is it worth designing family based software on it? For this, we create an economic model for it. This activity employs a group of domain engineers who currently produce PPS systems and a moderator who has got experience in FAST process. To qualify the PPS domain, we include 3 steps here:

*1.* *Estimated effort without domain engineering* - Our domain engineers estimate that the effort of producing a new member of the PPS family without the domain engineering is approximately 75 staff days for the average new member. Below includes all the critical tasks and their costs associated with software development.

|  |
| --- |
| Activity Cost |
| Understanding the new requirements 7 staff days |
| Understanding the design implications of 7 staff days  the new requirements |
| Changing the code ( with debugging ) 36 staff days |
| Unit testing on software simulator 15 staff days |
| Documenting 6 staff days |
| Acceptance testing 4 staff days  by the biologist |
| Total Effort 75 staff days |

*2.* *Estimated effort with domain engineering* - With domain engineering, our team estimates that we can cut the costs almost by a quarter. By partially automating the following activities, we can generate the code and documentation for PPS family members.

· Understanding the design implications, coding, and debugging

*Savings* – 20 staff days

· Unit Testing

*Savings* – 4 staff days

· Documenting

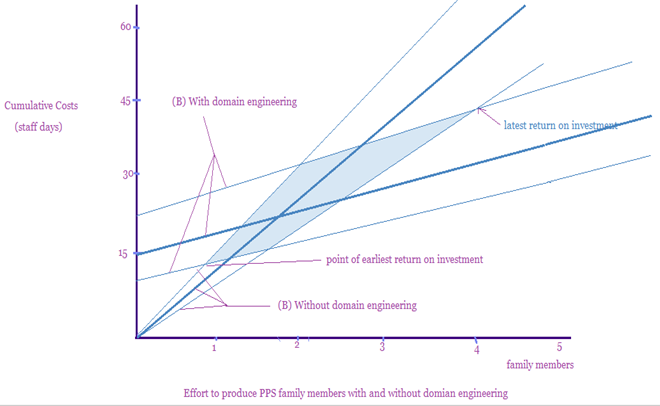
*Savings* – One staff day

Also, our group considers whether we add new activities to the process of producing new simulator. If we follow the FAST paradigm completely, our requirements for PPS will be expressed in a new language called OOPS and we guess that it may make it somewhat easier to understand the requirements. We believe that generated documentation may require some review and revision, so we allocate five days for this job. This reduces our effort savings to about 50 staff days as shown in the table below.

|  |
| --- |
| Activity Cost |
| Understanding the new requirements 6 staff days |
| Coding and debugging 24 staff days |
| Unit testing on software simulator 11 staff days |
| Reviewing & reviving general documentation 5 staff days |
| Acceptance testing by the biologist 4 staff days |
| Total effort 50 staff days |

We can observe considerable savings in effort estimation when domain engineering is used.

3. *Constructing an Economic Model* -- We can now construct a simple economic model for the PPS domain. The three lines labeled A show the effort to produce PPS family members with our current process. We use a range of lines to indicate the uncertainty in our estimate. The bold lines are our nominal cost estimates. The thin lines indicate the upper and lower bounds for our estimates. Similarly, the three lines labeled B indicate the effort we expect once we have engineered the domain, i.e., they include the cost of producing our application engineering environment. The shaded area represents the area in which we start to obtain a return on our investment in domain engineering. The return may come as early as the first PPS family member we produce with our application engineering environment, or as late as the fourth family member. We note that on the nominal lines we get a savings in effort of about 33% (50 staff days from 75) for each PPS system that we produce.

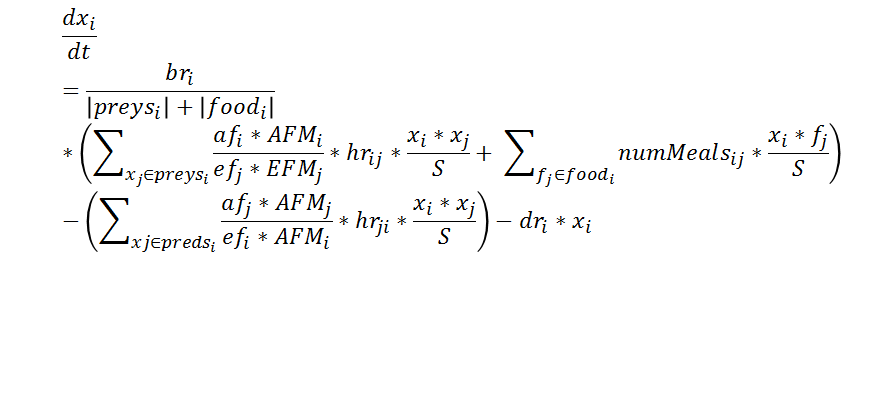


Our team summarizes the results of the domain qualification in a business case that identifies the biologists needs, the resources needed to do the domain engineering, and the return on the domain engineering.

## Dictionary of Terms and Equations

This section defines the list of terms and definitions as well as all assumptions and models that will be used to design the PPS software family.

* ***PPS*** Acronym for Prey-Predator Simulations
* ***OOPS*** Acronym for Object-Oriented Predation Specification, which is the name for the application modelling language designed in this document.
* ***Population change of species*** Defined by the following differential equation below assuming that there is no distinction between prey and predator. Refer to Table 1 below for the definitions of the variables in the formula.





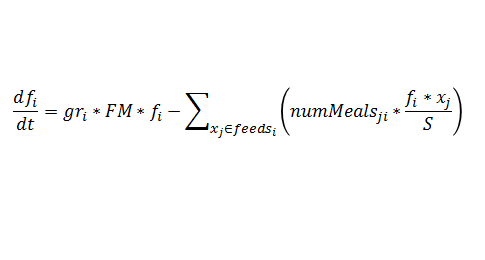
* ***Death rate of species*** Defined by the following equation below, assuming that disease is either 0 or 1. This also assumes that the rate of death doubles when disease is taken into account. Refer to Table 1 below for the definitions of the variables in the formula.



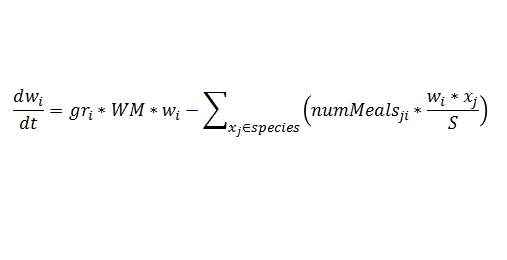
* ***Aggressivity and Evasiveness factor multipliers (AFM, EFM)*** Defined by the following equation below assuming that there is no distinction between prey and predator. This means every habitat has an evasiveness factor multiplier and an aggressiveness factor multiplier, which applies to all species. Refer to Table 1 below for the definitions of the variables in the formula.



* ***Change of solid food source*** Defined by the following differential equation below assuming that there is no distinction between the food sources. Refer to Table 1 below for the definitions of the variables in the formula.



* ***Change of water quantity*** Defined by the following differential equation below taking into account the land area and natural habitat. Refer to Table 1 below for the definitions of the variables in the formula. Please note that the PPS assumes every time the species eats, it will also drink.



|  |  |  |
| --- | --- | --- |
| Legend | Meaning | Example |
| xi | population size of species *i* | xsnakes refers to the population size of snakes at time t. |
| fj | size of food *j* | fcarrots refers to the amount of carrots available at time t. |
| wi | size of a water source *i* | wlakerefers to the water from a lake |
| numMealsij | The number of meals species *i* eats of food *j*. | numMealsrabbits,carrots=5\*t (5 times per day\* |
| bri | birth rate of a species *i* | brsnakes is the birth rate of snakes |
| gri | growth rate of [species, water, or food] *i* | grrabbits= xbunnies**\***14\*t (14 bunnies are born per available bunny per day) |
| dri | natural death rate of species *i* | drbunnies = xbunnies\*0.01 (1% of its population dies naturally) |
| afi | aggressive factor of species *i* | afsnakes = 1. |
| efi | evasiveness factor of species *i* | efrabbits = sqrt(2) |
| agingi | death rate of species *i* caused by aging | agingrabbits = 2/day |
| diseasei | a species *i*’s resistance to diseases. | diseaserabbits=0 resistance to all diseases; their death rate will double. |
| AFMi | aggressiveness multiplier of species *i* | AFMwolfs = sqrt(2) (they hunt in packs) |
| EFMi | evasiveness multiplier of species *i* | EFMwolfs = 1 (evasiveness factor not affected) |
| WM | water multiplier for all water sources | in Spring, WM = 2 |
| FM | food multiplier for all food sources | in summer, FM=2 |
| habitatAFM/EFM | the AFM or EFM associated with a given habitat | prairieAFM = 1; it does not affect aggressivity factor at all. |
| hrij | the rate at which species *i* hunts species *j* | hrsnakes,rats=3 |
| S | Scaling factor, includes land area (acres) and nature of habitat | S has a higher value (less interactions) for higher values of acre. S has a higher value for more difficult-terrain style habitats. |
| preysi | The set of all species that species *i* preys on | preyssnakes= {insects, mice, rats, birds, eggs, fish, frogs, lizards, …} |
| foodsi | The set of all non-hunting food that species *i* eats | foodsnakes = {}; snakes are carnivorous and must predate for food. |
| predsi | The set of all species that predates on species *i* | predssnakes={owls, hawks, falcons, heron} |
| |preysi| | Cardinality of the set of all preys species *i* preys on. | |preysrabbits| = zero; rabbits do not prey on any species. |
| |foodi| | Cardinality of the set of all non-hunting food species *i* eats. | |foodsnakes| = zero; snakes are carnivorous and must predate for food. |
| feedsi | The set of all species that feeds on food *i* | feedscarrots= {rabbits} |
| species | The set of all species in the PPS | species = {rabbits, snakes] |
| water | The set of all water sources in the PPS | water={lake, spring, river} |

**Table 1 - Legend for the above three differential equations**

## Scoping Commonality and Variability Analysis

**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 resistance. | Specification | 0 |

## 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. |

## Decision Model For Prey and Predator

|  |  |
| --- | --- |
| **Decision** | **Order (Phase)** |
| Define Interactive coefficient (Aggressiveness factor) | 1 |
| Define Interactive Coefficient(Evasiveness Factor) | 1 |
| Maximum RainFall | 2 |
| Minimum RainFall | 2 |
| Highest Vegetation Growth Rate | 1 |
| Lowest Vegetation Growth Rate | 1 |
| Determine the Clean Water Rate | 2 |
| Determine the Land Area(# in Acres) | 1 |

## Family Design Modules

The PPS software family module structure is based on the decomposition criteria of information hiding. The purpose of this section is to document all modules of the software family and the services and secrets they contain. This is simply an attempt to design the PPS in such a way as to minimize the expected cost of software and likelihood of changes when specific applications are developed under the family.

### FD1. Graphical User Interface Modules

The graphical user interface (GUI) modules consist of all the GUI outputs that the user should be able to see in an application and all the GUI controls in order to provide the inputs to a simulation.

**Plot View Module**

***Service:*** Provides a graphical representation of the size of the populations of all the species involved in the simulation, quantity of food sources, and quantity of water every week.

***Secret:*** How to display calculated values in the form of a plot.

***Associated variabilities and parameters of variation:*** V2 Season, this may be noted in the x-axis of the plot. V4 Disease, epidemics may be visualized in the plot view.

**Input Controls Module**

***Service:*** Provides ability for the user to give inputs to the application. Controls will be on the following:

* Habitat drop-down selection
* Species checkbox selections
* AFM, EFM, WM, FM input textbox values
* Season drop-down selection
* Hunting style, disease resistance checkboxes for each specie.
* Birth rate, death rate drop-down selections for each specie.
* Aggressiveness and evasiveness factors textbox input for each specie.
* Hunting rate, number of meals, initial population size textbox input for each specie.
* Preys and food sources for each specie checkboxes.

***Secret:*** How to extract the values from the input controls such as textboxes, checkboxes, and drop-down selections.

***Associated variabilities and parameters of variation:*** V1, V2, V3, and V4

### FD2. Software-Design-Hiding-Modules

The software-design-hiding modules hides software decisions that directly impact programming and algorithmic decisions as well as coding for performance.

**Population Size Calculation Module**

***Service:*** Provides an algorithm for determining the size of the population of a specie.

***Secret:*** How to implement the ***Population change of species*** formula using programming code.

***Associated variabilities and parameters of variation***: V1, V2, V3, and V4

**Solid Food Source Size Calculation Module**

***Service:*** Provides an algorithm for determining the size of the food source.

***Secret:*** How to implement the ***Change of solid food source*** formula using programming code.

***Associated variabilities and parameters of variation***: V1, V2, V3, and V4

**Water Size Calculation Module**

***Service:*** Provides an algorithm for determining the quantity of water in habitat.

***Secret:*** How to implement the ***Change of water quantity*** formula using programming code.

***Associated variabilities and parameters of variation***: V1, V2, V3, and V4

## OOPS Application Modelling Language (AML)

This section describes the application modelling language (AML) for the PPS family. This AML defines a specification language intended for the user of the software to specify requirements and for the application engineer to understand those requirements and easily implement them.

The AML defined here is called Object-Oriented Predation Specification, otherwise known as OOPS. It is inspired by the object-oriented syntax of the C# programming language, but is not to be confused with the actual C# syntax, and is composed entirely of OOPS Structures.

**Using OOPS Structures**

OOPS Structures are used to easily specify all the characteristics of each habitat such as: AFM, EFM, WM, FM, list of all seasons possible in the habitat, and the list of all species in the habitat. Below is the syntax for specifying the above information.

struct NameOfHabitat

{

// AFM = Aggressiveness Factor Multiplier

const double AFM = ##;

// EFM = Evasiveness Factor Multiplier

const double EFM = ##;

// WM = Water Multiplier

const double WM = ##;

// FM = Food Multiplier

const double FM = ##;

// List of all possible seasons experienced by the habitat.

List<> Seasons;

Seasons.Add(“NameOfSeason1”);

Seasons.Add(“NameOfSeason2”);

Seasons.Add(“NameOfSeason3”);

...

// List of all species in the habitat.

List<> Species;

Species.Add(“NameOfSpecie1”);

Species.Add(“NameOfSpecie2”);

Species.Add(“NameOfSpecie3”);

...

}

Where:

* The keyword struct refers to the word “structure”.
* The // indicates a comment. This is optional and may be used to explain the reason behind the next OOPS statements. Comments may also be used to ask questions about how to specify other quantities, so that the domain engineers could take in feedback to improve the OOPS AML.
* The name NameOfHabitat is substituted for the actual habitat name.
* The keyword const refers to the word “constant”, where once specified in the structure, the value cannot change.
* The keyword double refers to the fact that the value provided is not a whole number. It is expected that the value will have a decima point.
* AFM, EFM, WM, and FM refers to Aggressive Factor Multiplier, Evasiveness Factor Multiplier, Water Multiplier, and Food Multiplier, respectively.
* List<> and Add() is used to create and add to a list as is done for the Seasons and Species. The names “NameOfSeason1” and “NameOfSpecie1” and the other sequential names are substituted for the actual names of the seasons and species, respectively. The names must be supplied in Pascal casing (first letter of word is capitalized).
* Note 1: a semicolon must be included at the end of each statement within the structure.
* Note 2: Names of the seasons and species within the Add() must be wrapped with quotation marks.
* Note 3: it is assumed that you will create multiple OOPS Structures to specify multiple habitats.

Below is Example 1 of a habitat structure with stubbed values of AFM, EFM, WM, and FM for the sake of this example.

struct Meadows

{

const double AFM = 0.25;

const double EFM = 0.75;

const double WM = 1.10;

const double FM = 1.20;

List<> Seasons;

Seasons.Add(“Winter”);

Seasons.Add(“Spring”);

Seasons.Add(“Summer”);

Seasons.Add(“Fall”);

List<> Species;

Species.Add(“Cheetah”);

Species.Add(“Deer”);

Species.Add(“Lion”);

Species.Add(“Gazelle”);

}

**Example 1: Example of use OOPS AML to define Meadows habitat**

We can use another OOPS structure to specify the characteristics and relations between other species for each specie. In this structure we specify the following for each specie (predator): hunting style, ability to resist disease, and the list of all species that this specie (predator) considers as prey. Here is the syntax that specifies this information.

struct NameOfPredator

{

// Style of hunting. 0 = predator hunts alone. 1 = predator hunts

// in pack.

bool HuntingStyle = ##;

// How predator copes with disease. 0 = no resistance. 1 =

// resistance to disease.

bool Disease = ##;

// Birth and death rates of predator.

const double BirthRate = ##;

const double DeathRate = ##;

// Aggressiveness and evasiveness rates of predator.

const double AggressivenessFactor = ##;

const double EvasivenessFactor = ##;

// The hunting rate of the predator.

const double HuntingRate = ##;

// Number of meals in a day.

const int NumberOfMeals = ##;

// Initial population size.

const int InitialPopulationSize = ##;

// List of all species that the predator feasts on.

List<> Prey;

Prey.Add(Species1);

Prey.Add(Species2);

Prey.Add(Species3);

...

}

Where:

* The keyword struct refers to the word “structure”.
* The // indicates a comment. This is optional and may be used to explain the reason behind the next OOPS statements. Comments may also be used to ask questions about how to specify other quantities, so that the domain engineers could take in feedback to improve the OOPS AML.
* The name NameOfPredator is substituted for the actual predator name.
* The keyword bool refers to the word “boolean”, where only 2 values are allowed 0 or 1.
* List<> and Add() is used to create and add to a list as is done for the list of Prey. The names Species1, Species2, and so on are substituted for the actual names of the preyed species. The names must be supplied in Pascal casing (first letter of word is capitalized).
* Note 1: a semicolon must be included at the end of each statement within the structure.
* Note 2: Names of the prey within the Add() must be wrapped with quotation marks.
* Note 3: it is assumed that you will create multiple OOPS Structures to specify characteristics and relations of multiple predators.

Below is Example 2 of a predator structure with constant values stubbed for the sake of this example. It also uses information from the Meadows habitat structure defined in Example 1 assuming that we define one of the species from the Meadows habitat structure.

struct Cheetah

{

bool HuntingStyle = 0;

bool Disease = 1;

const double BirthRate = 3.0;

const double DeathRate = 1.0;

const double AggressivenessFactor = 3.0;

const double EvasivenessFactor = 2.0;

const double HuntingRate = 2.0;

const int NumberOfMeals = 2;

const int InitialPopulationSize = 250;

// Meadows.Species[1] is equivalent to Deer from Example 1.

// Meadows.Species[3] is equivalent to Gazelle from Example 1.

List<> Prey;

Prey.Add(Meadows.Species[1]);

Prey.Add(Meadows.Species[3]);

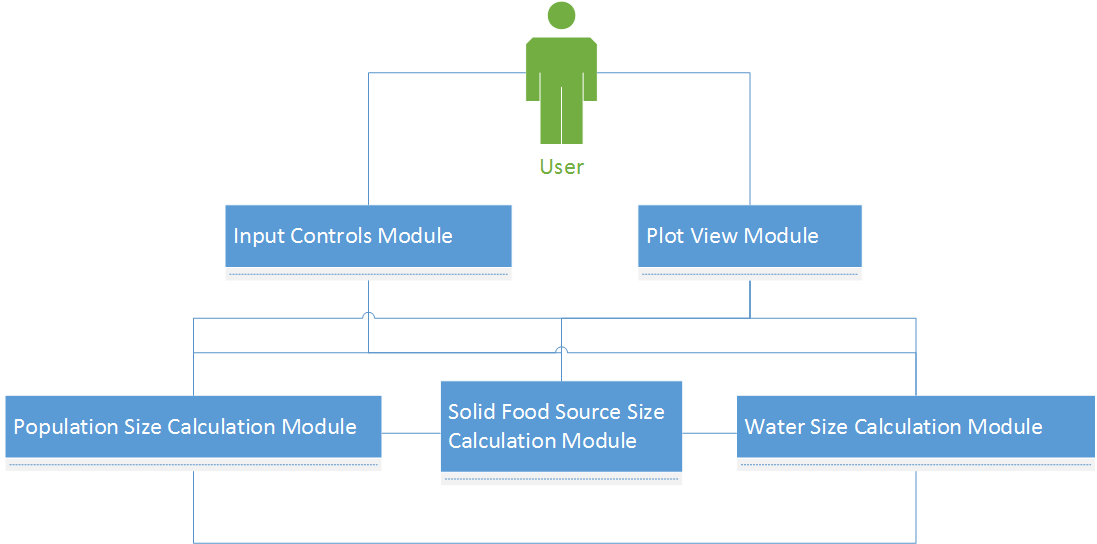
}

**Example 2: Example of predator structure for Cheetah using the Meadows structure from Example 1**

In Example 2, notice how Meadows.Species[1] is specified instead of simply “Deer”. This is where the “Object-Oriented” part of OOPS comes into play and is useful for the development engineers to understand that values specified earlier can be reused again in the new structure in AML.

## Uses Relations And Composition Mapping

This section shows the Uses Relation among the PPS modules designed in the Family Design section.



The below table shows the composition mapping between the decision and modules used to implement the decision:

|  |  |
| --- | --- |
| **Decision** | **Modules(s)** |
| Define Interactive coefficient (Aggressiveness factor) | Population Size Calculation Module |
| Define Interactive Coefficient(Evasiveness Factor) | Population Size Calculation Module |
| Maximum RainFall | Water Size Calculation Module |
| Minimum RainFall | Water Size Calculation Module |
| Highest Vegetation Growth Rate | Solid Food Source Size Calculation Module |
| Lowest Vegetation Growth Rate | Solid Food Source Size Calculation Module |
| Determine the Clean Water Rate | Water Size Calculation Module |
| Determine the Land Area(# in Acres) | Population Size Calculation Module, Solid Food Source Size Calculation Module, Water Size Calculation Module |

## References

* Software Product Line Engineering: A Family Based Software Development Process. David M. Weiss and Chi Tau Robert Lai. Addison Wesley, 1999.
  + Chapter 5 An Example: The Floating Weather Station Family, Pages 67-92
  + Addendum A The Floating Weather Station Commonality Analysis, Pages 113-119
  + Addendum B The Floating Weather Station Module Guide, Pages 121-126
* <http://www.phschool.com/atschool/phbio/active_art/predator_prey_simulation/index.html> which is a legacy prey-predator example application to find out more about the product line.
* <http://www.eduweb.com/portfolio/studyworks/predators8a.html> which is another legacy prey-predator example application to find out more about the product line.
* <https://www.google.com/url?sa=t&source=web&rct=j&url=https://wr.informatik.uni-hamburg.de/_media/teaching/sommersemester_2013/paps-1213-betke-preypredatorsimulator-ausarbeitung.pdf&ved=0CFEQFjAPahUKEwiw75OlpPDHAhXBND4KHWqXBwE&usg=AFQjCNFf0Hwn7ZKEEq0o-kWbcmO0UTiVOA&sig2=uNhMESak2Yske73thD1wKg> which is a lab report on a prey-predator simulator used to gain knowledge of the model and software.
* <http://web.mit.edu/12.000/www/m2012/finalwebsite/problem/climate.shtml#precip> for water growth rates based on seasons.
* Class lecture notes for all the formulas and coefficients.