

POSEIDON PERL CODE GENERATION PLUGIN GUIDE

General Rules

- The result of the code generation is saved as Module file (*ClassName.pm*)
- Interfaces and their associations are not translated into Perl code
- Abstracts and their associations are not translated into Perl code
- Element's documentation are translated into Perl comment syntax (*# comment*)
- Attribute / Parameter types are ignored because there is no need to define data types for Perl variables

Classes

- Uses the standard UML 'Class'
- Classes are translated into Perl Class (package *className*)
- A Constructor is generated for each class (sub *new*)

Class Attributes

- Attributes are translated into variables
- Attributes with single multiplicity are translated into scalar type variables (*my \$AttributeName*)
- Attributes with multi multiplicity are translated into array type variables (*my @AttributeName*)
- An attribute with a tagged value 'local' set to 'true' is translated into 'local *\$AttributeName*' instead of '*my \$attribute*'
- An attribute that has non-1 multiplicity with a tagged value 'Map' set to 'true' is translated into '*%AttributeName*' instead of '*@AttributeName*'
- When the visibility of an attribute is public, 'use vars qw (*\$AttributeName*)' is added to the code generation.

Class Operations

- Operations are translated into subroutines (*Sub OperationName*)
- Parameters are translated into subroutine variables
- Return values are not translated into Perl code
- When an operation is static and has the stereotype *<<create>>* a 'BEGIN { }' block is added to the code generation.
- When an operation is static and has the stereotype *<<destroy>>* an 'END { }' block is added to the code generation.

Associations

- 1 to 1 associations are translated into scalar type variables (`my $className`)
- 1 to N associations are translated into array type variables (`my @className`)

Aggregation

- 1 to 1 aggregations are translated into scalar type variables (`my $className`)

Inheritance

- single inheritance is implemented using `'@ISA = qw(class)'`
- multiple inheritance is implemented using `'@ISA = qw(class1 class2 ...)'`