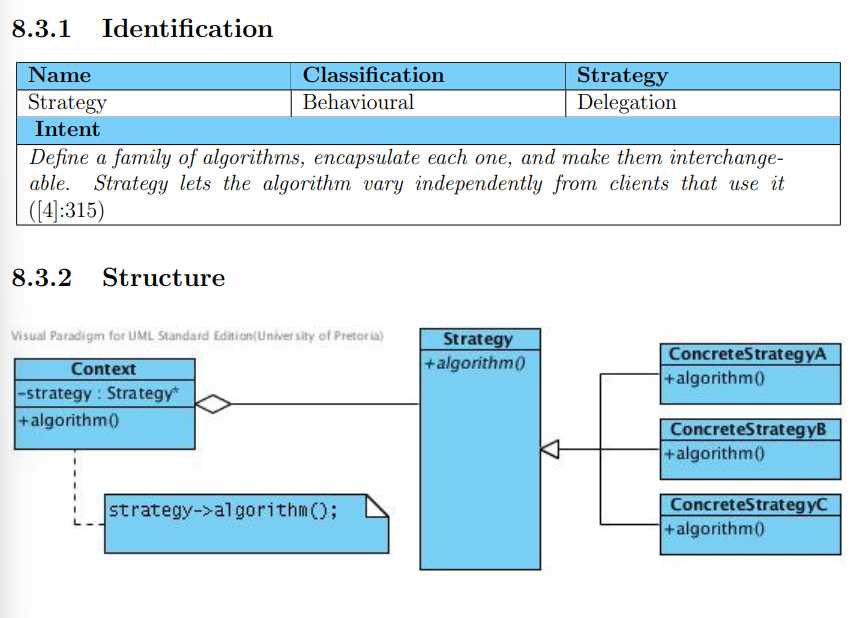
**Background**

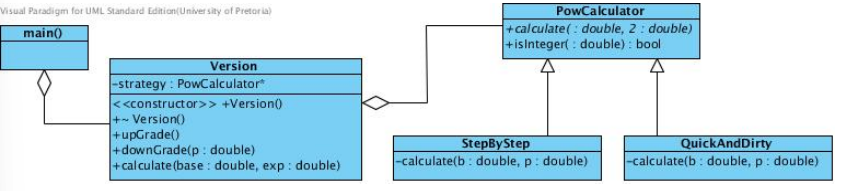
* If changing one module in a program requires changing another module, then coupling exists.
* It is practically impossible to avoid coupling. It can, however, be controlled.

**Strategy Design Pattern**

****

* Problem
  + Uncontrolled coupling
  + The coupling between the Strategy interface and the Context class might be wider than always needed
* Participants
  + Strategy
  + ConreteStrategy
  + Context
  + Client
* Improvements
  + Where various classes provide different implementations of the same routine, interface details is encapsulated in a base class, while the implementation details are provided in derived classes
  + Eliminates need for complicated conditional statements
  + Client doesn’t know about all the complex stuff
  + More robust
* Practical
  + Save files
  + GPS
  + Input fields
  + Line-breaking strategies to displat textual data
  + Network traffic
* Implementation issues
  + Selecting most appropriate strategy
* Misconceptions
  + When the different strategies are simple methods implemented without encapsulating them in classes that are polymorphic subclasses of an interface, the system implements the desired functionality but is not an application of the Strategy design pattern
* Related patterns
  + Factory Method
  + State
  + Flyweight

**Example**

****

**Table

Description automatically generated**