Interpreter Design Pattern [Gamma et al]

This design pattern allows the developer to build Classes as rules. The pattern focuses on defining formal language grammars (problem characterization) as domain language, domain rules as sentences and interpreting these sentences to solve the problem. Each grammar rule is represented by a class and the domain modeling is represented by recursive grammar. Each rule can be either a Composite (rule incorporate other rules) and terminal (a leaf node). This pattern implementation relies on the recursive traversal of Composite Pattern to interpret sentences. Following are the variants of interpreter pattern:

Single Interpretation [59]

This variant is simplest and quickest way of implementing visitor based interpreter. It deals with the interpretation of elements which holds domain interest. The points in the interpretation code where visitation occurs is called the points of visitation and the points where artifacts are generated are points of generation. Following is the conceptual model:

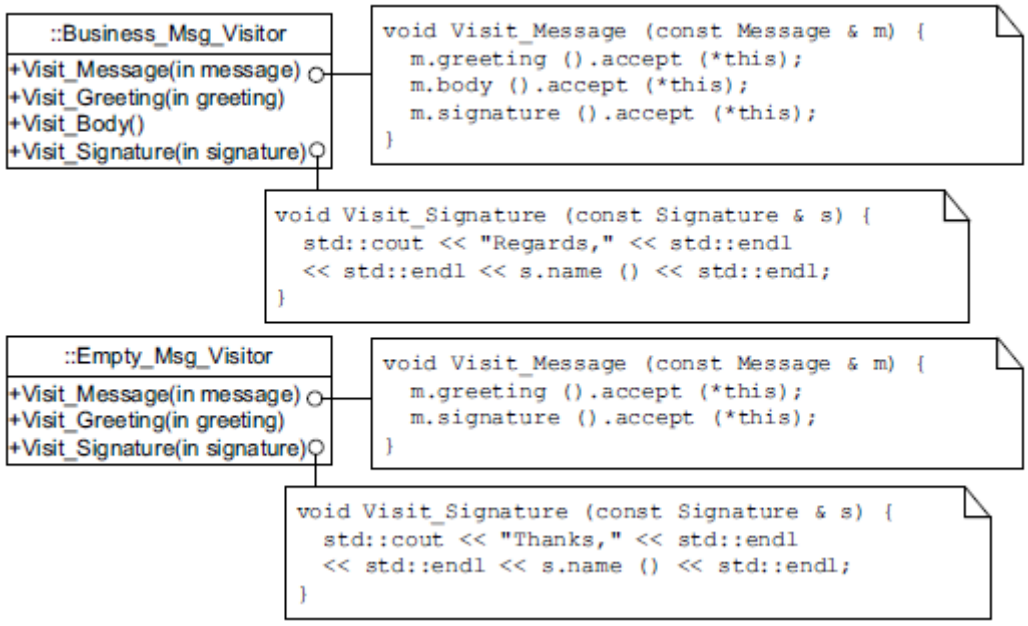

Figure 1.5 Conceptual model of Single interpreter's implementation

Strategized Interpretation [59]

This variant is a technique to address reinvention in the core interpretation logic partially when targeting multiple metadata formats. This variant is built on the top of Strategy Pattern [1]. Following is the conceptual model of the strategized interpreter:

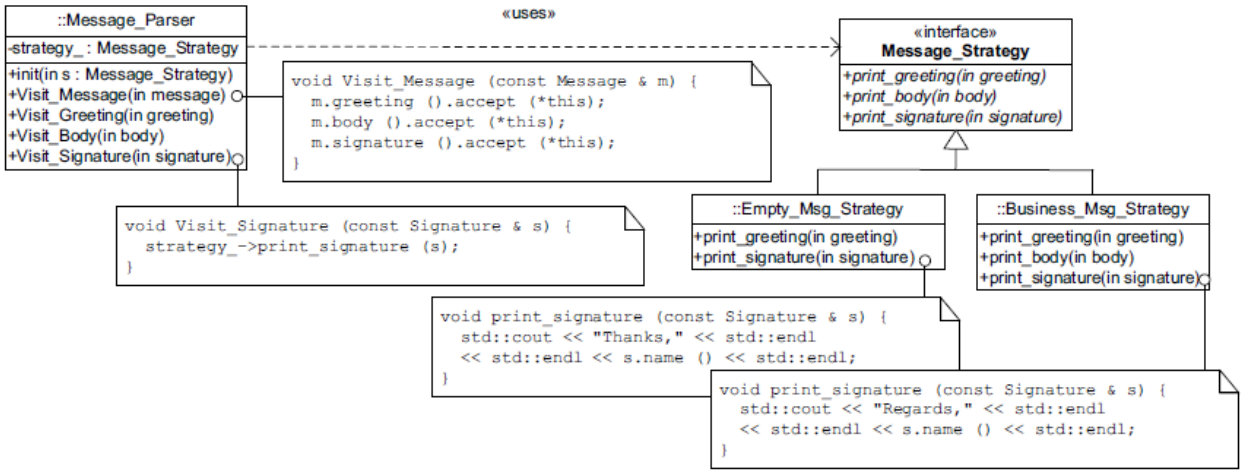

Figure 1.6 Conceptual Model of Strategized Interpreter