オブジェクト指向設計におけるデザインパターンの特定分野への適用に関する研究

清水裕光

北陸先端科学技術大学院大学 情報科学研究科

1996 年 2 月 14 日

キーワード: オブジェクト指向設計、再利用、デザインパターン、フレームワーク.

オブジェクトによって現実の世界を抽象化し、ソフトウェアを設計するオブジェクト指向設計の利点として部品の再利用が容易であるといわれている。しかし、オブジェクト指向に基づいて設計する際には、まずシステムを適切なオブジェクトに分割し、適切な粒度でクラスとしてまとめる。次にクラスのインタフェースや継承構造、そしてそれらの間の主要な関係を確立する必要がある。そのため、実際に再利用可能なソフトウェアを設計することには多くの困難を伴う。

オブジェクト指向設計における再利用を目的とし、多くの事例の中で繰り返し用いられている設計を体系化したものはデザインパターンと呼ばれている。しかしながら、汎用部品としての個々のデザインパターンの有用性はある程度実証されているが、デザインパターンは抽象度が高く、実際のシステム設計に適用するための方法に関しては、十分な研究がなされていないのが現状である。そこで本研究では、特定分野のアプリケーションのための再利用可能なコンポーネント (フレームワーク) の構築を例とし、デザインパターンの適用に関する事例研究を行い、再利用可能な設計に伴う問題点を検討する。

事例としては、言語処理系を考える。言語処理系を選んだ理由は、設計のための理論が確立しており、さまざまな分野への応用範囲が広いものであるからである。対象とする言語処理系は、字句解析器、構文解析器といったサブシステムから構成される。

言語処理系のためのフレームワークを設計する上で、特に考慮しなければならない点として以下のような問題について議論を行う。(1) 文法の変更容易性: 文脈自由文法の各文法記号は、終端記号と、非終端記号の二種類あり、それぞれ特有の性質を持っている。このことを考慮し、文法の変更を容易にできるようにする。(2) 出力の柔軟性: 構文解析器の出力は、一般には解析木や抽象構文木である。これらのノードは、言語によって任意に変化するが、解析木の作成の方法は一定である。よって、解析木はどのようなノードを作成する場合でも、一定の方法で行えるようにしなければならない。また抽象構文木の作成にあ

たっては、言語で扱うノードの種類を変えずに、木の作り方を変更するといった要求も起こり得る。したがって、木の作成法を容易に変更できるようにしなければならない。(3) 意味規則の利用: 文脈自由文法で表すのが難しい静的意味を定式化するための方法として、代表的なものに属性文法がある。一般に属性文法では、属性はさまざまな型をとり得る。したがって、フレームワークを設計するにあたっては、属性の型を容易に変更できるようにしなければならない。

言語処理系に対する要求分析の結果、フレームワークの構成について、以下のようなデザインパターンが利用可能であることが判明した。(1)Interpreter パターンを用いることによって、終端記号と、非終端記号のそれぞれの性質を定義したクラスを構築し、文法記号を表すクラスはこれらのクラスを継承して定義することによって、文法の変更を容易にし、構文解析を行って解析木を出力し、意味解析を行う方法を提供した。(2)Builder パターンは、抽象構文木の作成部を文法記号を表すクラスから分離し、抽象構文木の作成方法の変更を容易にした。また、意味解析で扱うクラスを抽象化することによって、属性を表す型の変更が容易になることを示した。(3)Iterator パターンによって、属性を表すリストや、解析木、抽象構文木といった集約オブジェクトへアクセスしたり走査する責任を分離し、集約オブジェクトの内部構造を明かすことなく走査し、集約オブジェクトの内部構造を変更することなく、走査方法の変更を可能にした。

本研究で構築したフレームワークを構成する抽象クラスを継承し、言語処理系のアプリケーションを実装することによって、フレームワークが十分な設計であるかどうかを検証するために、簡単な言語を例として、フレームワークを利用したアプリケーションの設計、実装を行った。その結果、構築したフレームワークを適用することによって、フレームワークを適用しないで構築した場合と比べて、容易に実装を行うことができることを示した。特に構文解析器は、文法を機械的に変換することによって、構築可能であることが判明した。その結果フレームワークは、言語処理系のアプリケーションを構築する上で必要とされる設計情報の大半を把握していることが分かった。

デザインパターンを適用して実際のシステムを構築する場合、対象となるシステムの設計問題に適しているパターンを選択し、そのパターンを使うことによる結果や、設計上のその他の制約を考慮した上でも適用できるかどうかを十分検討する必要がある。そのため、デザインパターンを特定分野に適用した例について、詳細に記述した文書は、同じようなシステムを構築する際に多いに有用である。さらに、デザインパターンでは触れられていない問題、例えば、構文解析中に文法が動的に変化するような場合の文書化は、今後待たれる課題である。