歯科診療報酬請求事務システムの開発
– 歯科診療システムとの統合を目的としたアーキテクチャ設計 –

2007MI172 丹羽 恵美　　2007MI236 田中 智紗衣
指導教員　沢田 篤史

1　はじめに

われわれは歯科診療報酬請求事務システム（以下，事務システム）の開発をおこなう．事務システムは，厚生労働省が発行しているマスタや仕様 [4] を基に設計と実装をおこなわなければならない．マスタには診療報酬制度により決定された診療情報の事務処理において参照するデータ群が記述されている．

診療報酬制度は定期的に改定され，それに伴いマスタの仕様が変更される．変更内容は処理内容の構成要素の組み合わせの変更と，構成要素内の一部の処理内容の変更がある．これらの変更により事務システムの処理を変更しなければならない．マスタに依存する処理の仕様は自然言語で記述されているので，技術者はマスタの仕様の変更を自然言語から解釈し直接処理を変更する必要がある．事務システムの作成にあたり，マスタや仕様の変更に対し処理内容変更の柔軟性を保証しなければならない．

本研究の目的は事務システムの処理部分（以下，処理系）を自動生成する生成系を作成し，技術者の処理内容の変更にかかる工数を削減することである．われわれは処理内容の柔軟性を保証するために，処理プログラムの自動生成を可能とするソフトウェアアーキテクチャを構築する．処理系のデータ構造はオブジェクト指向に基づき定義する．処理系には，マスタや仕様書の変更に対する保守性を保証するために GoF デザインパターン [1] の Visitor パターンを適用し，定義したデータ構造から事務処理を局所化する．データ構造には言語の文法表現とその解釈処理を実現する Interpreter パターンを適用する．これらのデザインパターンを適用することで，マスタや仕様の変更に対する処理系の処理内容変更の柔軟性を高める．

生成系の作成にあたり，処理の種類の観点から事務処理の部品化をおこなう．部品化した処理の中で，マスタの項目に含まれる識別子によって処理内容が変更される部分において，識別子と処理の対応表を作成する．生成系には，必要なソースコード部品の属性を定義した言語と対応表を入力とし，Interpreter パターンを適用した言語処理系のアーキテクチャを設計する．生成系に必要なソースコード部品の属性を定義した言語と対応表を入力することで処理ロジックの変更と処理内容の変更に柔軟に対応することが可能となった．作成した処理系と生成系に適用するデザインパターンを可変性の実現方法に着目して考察した．処理系と生成系のアーキテクチャから，変更された処理内容の生成可能性について考察した．

2　歯科診療報酬請求事務システム

2.1　システムの概要

歯科診療システムは，診療記録システム，診療予約受付・清算システム，診療報酬請求事務システムの三つのシステムで構成されている．本研究では，歯科診療システムにおける診療報酬請求事務システムを作成する．歯科診療システムを構成するシステムと，その関連を図 1 に示す．

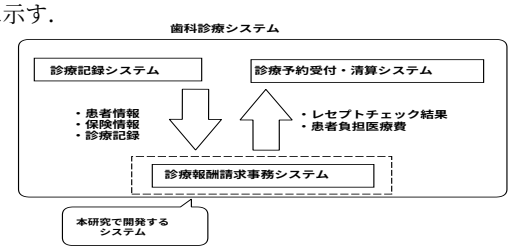

図 1　診療報酬システムと他のシステムとの関連

事務システムの主な機能は以下の三つである．

1. レセプトチェック
   レセプトに入力した診療記録が正しいものか，保険点数として算定可能かどうかを検査する．
2. 保険点数算定
   事務処理における患者負担医療費（保険点数）を算定する．
3. 電子レセプト作成
   月末に審査機関に提出する電子レセプトを作成する．

事務システム全体は，処理系と事務処理の処理内容を生成する生成系から構成されている．図 2 に事務システム全体のアーキテクチャを示す．

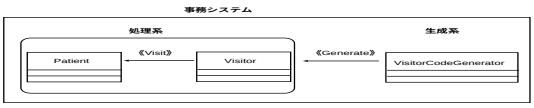

図 2　システム全体のアーキテクチャ

2.2　マスタファイル

事務システムの三つの機能は，厚生労働省が発行しているマスタや仕様を基に実装する．マスタには診療報酬制度により決定された，診療情報の事務処理において参照するデータ群が記述されている．マスタのデータには事務システムで扱う診療行為名称などの情報と，その情報の処理方法を定義するデータが存在する．マスタやマスタの項目の仕様は定期的に変更される．マスタやマスタの項目の仕様の変更に応じて，診療報酬の参考書など

に変更が生じる．変更内容には，既存の項目の変更・追加と，新規のマスタの項目の追加がある．これらの変更により，処理系の処理内容を変更する必要が生じる．これらのマスタの変更にも問題なく処理ができることが，事務システムの非機能要求として挙げられる．

2.3 GoF のデザインパターンの適用

処理系のアーキテクチャ設計には GoF のデザインパターンを適用した．GoF のデザインパターンとは，オブジェクト指向に基づく設計によく用いられるパターンに名前を付けてまとめたものである [1].

GoF のデザインパターンに 23 個のパターンがふくまれている．われわれは，処理系に言語の文法表現のその解釈処理を実現する Interpreter パターンとデータ構造に対する処理を局所化する Visitor パターンを適用した．

Interpreter パターンは患者情報のデータ構造の走査順序を定義している．処理系のデータ構造を走査する順序を定義し，事務処理を実現する．

Interpreter パターンを適用したデータ構造に Visitor パターンを適用し，事務処理の三つの処理を局所化する．事務システムの機能であるレセプトチェック，保険点数算定，電子レセプト作成は，Visitor のサブクラスで処理をおこなう．事務処理をデータ構造から分離し局所化することで，診療報酬の仕様の変更に対して柔軟な構造となる．図 3 に Interpreter パターンと Visitor パターンを適用した処理系の静的構造を示す．

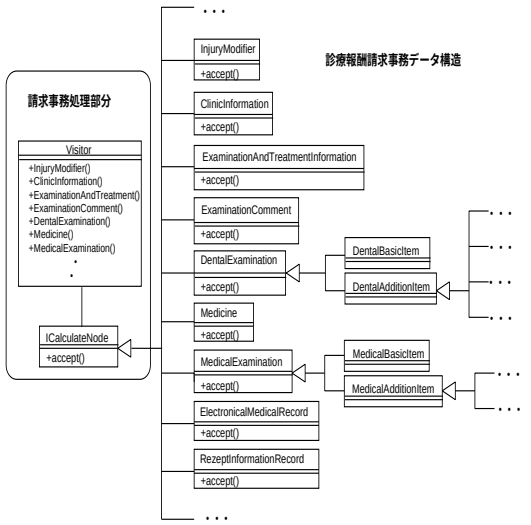

図 3　Interpreter パターンと Visitor パターン適用後の静的構造

3　歯科診療報酬請求事務処理の生成系の設計

3.1 生成系の概要

処理系の処理部分である Visitor のソースコードを生成する生成系の概要を図 4 に示す．

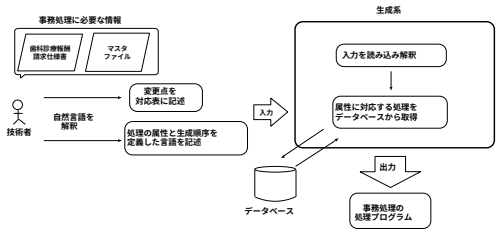

図 4　生成系の概要

生成系は事務システムの事務処理（保険点数算定，電子レセプト作成，レセプトチェック）の事務処理を生成する．生成系には Interpreter パターンで解釈する言語と対応表を入力とする．生成系は入力された言語を解釈し，データベース上に記録されているソースコード部品を取得することで必要なソースコードを生成する．

3.2 部品化した処理の生成の実現方法

生成系の作成にあたり，処理の種類の観点から事務処理の部品化をおこなう．部品化した処理を組み合わせて Visitor の処理内容を生成することで，生成順序の変更に柔軟に対応することができる．保険点数算定処理を例とした部品化した概念図を図 5 に示す．

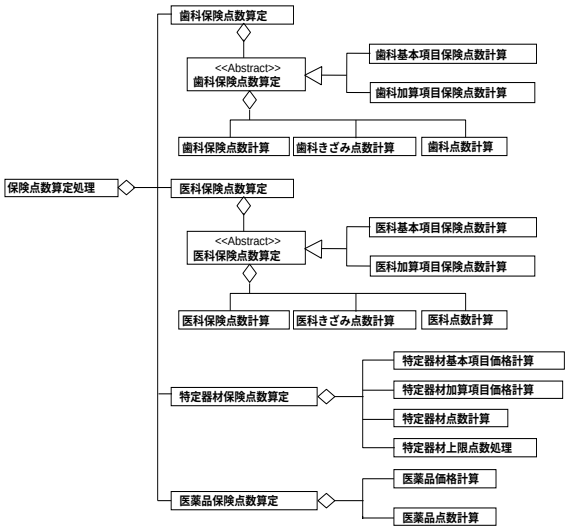

図 5　保険点数算定処理の部品化した概念図

部品化したそれぞれの処理を表す言語を定義し，言語を組み合わせ処理の属性と生成する順序を指定する．生成系に組み合わせた言語を入力することで指定した順序で処理を生成する．

3.3 識別子と処理の対応表

マスタの項目に含まれる識別子とそれに対応する処理は一対一対応している．マスタの項目に含まれる識別子とそれに対応する処理の対応表を作成する．対応表を用いることで技術者が直接プログラムコードを変更する工数を削減する．開発者は仕様書等に記載された自然言語

を解釈し，マスタの項目に含まれる識別子とそれに対応する処理をプログラムコードとして生成系に入力する対応表に記述する．

技術者は，マスタの項目に含まれる識別子と対応する，自然言語で記載された処理を解釈し，そのプログラムコードを表に入力する．表 1 に保険点数算定の歯科基本項目の項番 11 の識別子の対応表を示す．

表 1　保険点数算定の歯科基本項目の項番 11 の対応表

	項番 11 の識別子	処理
歯科基本の保険点数算定	1	Count += (Master12/10);
	3	Count += Master12;
	4	Count += MedicalInstitutionCount;
	7	Count -= MedicalInstitutionCount;
	8	Count -= Master12;

表の左側にマスターテーブルの項番に設定されている識別子の種類を記述し，表の右側には識別子に一対一対応している処理のプログラムコードを記述する．

3.4　GoF のデザインパターンの適用

生成系のアーキテクチャ設計に GoF デザインパターンを適用した．生成系には処理の属性と生成順序を表す言語を入力する．処理の属性と生成順序を表す言語の解釈処理を実現する Interpreter パターンと再帰的な構造を実現する Composite パターンを適用する．図 6 に保険点数算定 Visitor の生成系の静的構造を示す．

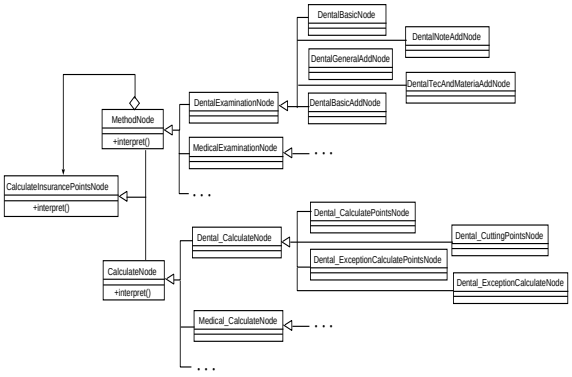

図 6　保険点数算定 Visitor の生成系の静的構造

処理の属性は木構造となるので Interpreter パターンと Composite パターンを適用して生成系に入力された言語を解釈する．

4　考察

4.1　可変性の実現方法に適用したデザインパターンの考察

ソフトウェアの再利用性を目的とした処理の変更に対する柔軟性を与える方法として，可変性の実現方法 [2] がある．歯科診療報酬請求事務システムで想定する変更に対して用いた可変性の実現方法は，多相型（Inheritance），テンプレート（Template Instantiation），構成（Configuration），生成（Generation）の四つである．四つの可変性の実現方法に適用するデザインパターンの妥当性について考察をおこなう．

4.1.1　処理系に適用したデザインパターンの考察

事務システムで想定する変更に対して用いた可変性の実現方法は多相型である．多相型が実現する可変性は，データ構造に対する処理内容の変更である．本研究では多相型に適用するデザインパターンとして，Visitor パターン，Strategy パターン，Chain of Responsibility パターンを考えた．これらのパターンに対し，処理内容の変更・追加・削除に対する処理内容の生成可能性と，データ構造の追加に対する処理の生成可能性の観点から比較をおこなった．比較をおこない，マスタや仕様の変更に対して柔軟に対応することができるアーキテクチャを構築するデザインパターンであるか妥当性を確かめた．Strategy パターンを適用した場合，処理内容の変更には柔軟だが，処理内容を変更する際に変更しなければならないクラスが複数に及び工数がかかることが欠点として挙げられる．Chain of Responsibility パターンは適用することで処理内容の変更が柔軟になるが，現在処理系の処理は一対一対応しており，要求に対する処理は静的に決定しているので，Chain of Responsibility パターンを適用する利益は少ないと考えている．Visitor パターンを適用することで，事務処理のデータ構造から処理を局所化し，処理内容の変更に柔軟に対応することができる．よって，多相型の実現方法として妥当なデザインパターンは Visitor パターンである．

表 2　多相型に適用したデザインパターンの比較

	処理の変更・追加・削除に対する処理内容の生成可能性	保険点数算定処理への有用性
Visitor	○	○
Strategy	△	○
Chain of Responsibility	○	△

4.1.2　生成系に適用したデザインパターンの考察

事務システムで想定する変更に対して生成系に用いた可変性の実現方法は，テンプレート，構成，生成の三つである．これらの実現方法に適用するデザインパターンの考察をおこなった．

テンプレートが実現する可変性は，マスタや歯科診療報酬請求の仕様書の変更により生じる事務処理の処理内容の変更・追加・削除である．テンプレートに適用するデザインパターンとして，Factory Method パターン，Abstract Factory パターン，Template Method パターンを考えた．これらのパターンに対し，処理内容の変更・追加・削除に対する処理内容の生成可能性と，生成系の保守性の高さの観点から比較をおこなった．比較をおこない，マスタや仕様の変更に対して柔軟に対応することができるアーキテクチャを構築するデザインパターンであるか妥当性を確かめた．Template Method パターンに関しては，データと処理が混在しているので，処理の変更に対して柔軟ではない．Factory Method パター

ンに関しては，事務処理の処理の種類が多数存在するので，処理内容に変更が生じた際に変更に工数がかかる．Abstract Factory パターンを適用することで，処理の部品ごとに処理内容を変更することができるので，処理内容の変更に対し柔軟に対応することができる．以上よりテンプレートの実現方法として妥当なデザインパターンは Abstract Factory パターンである．テンプレートに適用したデザインパターンを比較した表を表 3 に示す．

表 3　テンプレートに適用したデザインパターンの比較

	処理の変更・追加・削除に対する処理内容の生成可能性	生成系の保守性の高さ
Factory Method	○	△
Abstract Factory	○	○
Template Method	△	△

構成が実現する可変性は，マスタや歯科診療報酬請求の仕様書の変更により生じる事務処理の変更・追加・削除である．構成に適用するデザインパターンとして，Strategy パターン，Interpreter パターンを考えた．これらのパターンに対し，処理内容の変更・追加・削除に対する処理内容の生成可能性と，処理論理の変更の容易性の観点から比較をおこなった．比較をおこない，マスタや仕様の変更に対して柔軟に対応することができるアーキテクチャを構築するデザインパターンであるか妥当性を確かめた．Strategy パターンは処理論理の変更の際，メソッドに変更したい箇所を記述する必要があるので処理論理の変更が容易であるといえない．以上より，構成の実現方法として妥当なデザインパターンは Interpreter パターンである．構成に適用したデザインパターンを比較した表を表 4 に示す．

表 4　構成に適用したデザインパターンの比較

	処理の変更・追加・削除に対する処理内容の生成可能性	処理論理の変更の容易性
Strategy	○	△
Interpreter	○	○

生成の可変性の実現方法は，処理内容の生成である．生成に適用するデザインパターンとして，Interpreter パターンと Composite パターンの組み合わせを考えた．生成系は，マスタの項目に含まれる処理の識別子と対応する処理の対応表と，処理の属性と生成する順序を指定する言語を入力とする．入力された情報を基に，処理内容を生成する．処理内容の生成にあたり，入力された情報の解釈をおこなう必要がある．処理の属性は木構造となるので，Interpreter パターンを適用して解釈をおこなう．入力される情報は複数存在する可能性があるので，Composite パターンを適用して再帰的に解釈をおこなう．以上より，生成系に適用するデザインパターンとして Interpreter パターンと Composite パターンの組み合わせが妥当である．以上より，生成系に適用するデザインパターンとして，今回適用した Interpreter パターンと Composite パターンが妥当であるといえる．

4.2　生成可能性に関する考察

生成系を実現したことにより得られる利点を以下に挙げる．

1. 生成系の事務処理の生成により，直接 Visitor に手を加える必要がない
2. 技術者が扱う情報の削減

現在存在する診療報酬制度の改定によるマスタや仕様の変更の種類を以下に示す．

- 処理内容の構成要素の組み合せの変更
- 構成要素内の一部の処理内容の変更

処理内容の構成要素の組合せの変更に対しては，部品化した処理を表す言語を組み合せることで変更された処理を生成することが可能である．構成要素内の一部の処理内容の変更に対しては，マスタの項目に含まれる識別子と処理の対応表を書き換えることで変更された処理を生成することが可能である．以上より，生成系は現在考えられるマスタや仕様書の全ての変更に対して，処理内容を生成することが可能である．

5　おわりに

本研究では歯科診療報酬事務システムの処理内容の柔軟性を保証することを目的として，処理プログラムの自動生成を可能とするソフトウェアアーキテクチャの構築をおこなった．マスタや仕様の変更に対応した処理内容を生成する生成系を作成し，処理系の処理プログラムの自動生成を可能とした．処理系と生成系に GoF デザインパターンを適用することで処理内容の変更に柔軟な構造とした．考察として，処理系と生成系に適用したデザインパターンと他のデザインパターンとの比較をおこなった．比較をおこない，マスタや仕様の変更に対して柔軟に対応することができるアーキテクチャを構築するデザインパターンであるか妥当性を確かめた．

参考文献

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wealey, 1995.

[2] I. Jacobson, M. Griss, P. Jonsson, *SOFTWARE REUSE: Architecture, Process, and Organization for Business Success*, Addison-Wesley, 1997.

[3] 後藤洋，“ Java ソースコードの CDI(Code Inspection) の開発 ～アーキテクチャの構築～，” 南山大学大学院 数理情報研究科 2008 年度 修士論文要旨集，pp.130-133，March 2009.

[4] 厚生労働省保険局，“ 診療報酬情報提供サービス，” http:/www.iryohoken.go.jp/shinryohoshu/，2010.