詐欺抵抗力診断 Web アプリのオブジェクト指向設計*

小久保 温†

Object-Oriented Design of the Fraud Resistance Diagnostic Web Application

Atsushi KOKUBO

ABSTRACT

In this paper, I discuss the object-oriented design of the web application named "Professor Watanabe's Fraud-Resistance Diagnostic".

This application was released to the public on the Web, but was also used for research. In the case of research, it is difficult to predict the final state of completion at the beginning. It was anticipated that many changes would occur during development and operation. Therefore, object-oriented designs were utilized to develop the application that is resistant to change. The app allows multiple surveys to be created by freely combining questions, branches, and diagnostics. The algorithms used for diagnosis can also be arbitrary.

Key Words: *object-oriented design, flexible, for research use, web application*
キーワード: オブジェクト指向設計, 高い自由度, 研究用, Web アプリケーション

1. 詐欺抵抗力診断 Web アプリ

この論文では，詐欺抵抗力診断 Web アプリのオブジェクト指向設計について論じる．まず，その開発の経緯について紹介する．

科学技術振興機構(JST) 社会技術研究開発センター(RISTEX)の「安全な暮らしをつくる新しい公／私空間の構築」研究開発プロジェクトの一つに，「高齢者の詐欺被害を防ぐしなやかな地域連携モデルの研究開発」(研究代表者・渡部諭・当時秋田県立大学教授)が 2017 年度に採択された [1)]．プロジェクトでは，2017 年 10 月～2021 年 3 月の 3 年半の期間に，特殊詐欺被害の減少を目指して詐欺に対する抵抗力を診断する Web アプリ「わたなべ教授のサギ抵抗力しんだ～ん」を開発・運用し，アプリを活用して詐欺被害防止の啓発活動に取り組んだ [2)]．

アプリは，俗に「診断アプリ」と呼ばれるものの一種である．ユーザーはブラウザからアプリにアクセスして質問に回答していくと，どの種類の詐欺にどのくらい強いのか，逆に言えば弱いのかが診断されて表示される．診断には，機械学習の一種のロジスティック回帰が用いられていた．アプリは Web で 2019 年 2 月下旬から 2021 年 3 月まで公開された．

* 令和 5 年 12 月 1 日 受付
† 工学部工学科／大学院工学研究科電子電気・情報工学専攻・教授

2. アプリの開発の履歴と担当

アプリは大きく分けて次の5つのバージョンがある．筆者はアプリのプログラムの開発と運用の責任者であった．

1. 要件策定用
   2017 年に，プロジェクトで詐欺抵抗力診断アプリの振る舞いの仕様を決めるために，診断が動作する最小限のプログラムを JavaScript で筆者が開発した．振る舞いを検討できればよかったので，データは永続化する必要はなく，データベースを使用していない．また，プログラムもすべてクライアントサイド，つまりブラウザ上で動いていた．
2. 設計確認用
   プログラムの開発を委託することになった会社 A から，2018 年に仕様だけでなく設計を提供して欲しいと依頼され，筆者は設計を作成した．そして，設計通りに開発して問題ないかを確認するためにサーバーサイドで動作するプログラムを Ruby on Rails で筆者が開発した．このバージョン以降，データは関係データベースに保存している．このプログラムも設計と共に会社 A に提供した．なお，プログラムの画面は動作検証用の必要最小限のものであった．
3. 初年度公開版
   会社 A が 2019 年に PHP と Angular でプログラムを開発し，およそ 1 年間運用した．筆者が依頼されて提供した設計には沿っていないものだった．また，会社 A では，診断に用いるロジスティック回帰のプログラムを開発できないとのことで，この部分は筆者が PHP で開発したプログラムを使用した．
4. 2 年度以降公開版
   会社 A が開発した初年度公開版は，公開したところデータが保存されないことがあるという不具合が発生したが，会社 A は修正できずその後の開発も行わないとの連絡を受けた．そこで，2019 年秋から，元の設計に沿って筆者が Ruby on Rails でプログラムを開発しなおした．不具合が発生しないように，念のため質問の回答の処理が適切に行われているかチェックするプログラムを，グラフ理論を用いて実装している．このバージョンが公開された 2020 年の冬以降，不具合は発生しなくなった．画面の開発と運用だけ会社 B に業務委託した．
5. 管理機能付き
   2021 年に，Web 画面から診断を入れ替えられる機能をつけた．それまでは，データベースやソースコードを直接いじって診断を入れ替えていた．開発は会社 B に委託した．

最終的なアプリでは，会社 B が画面と管理機能を開発し，それ以外のすべてを筆者が開発している．この論文では，筆者が開発した範囲の最終的なアプリの設計について論じる．

3. アプリの仕様

アプリの仕様は次のように定めた．

3.1 目的

高齢者の詐欺被害を防ぐために，研究および予防・啓発を行う．そのために，高齢者を対象に詐欺脆弱性を診断する Web アプリを開発する．

3.2 エンド・ユーザー

アプリのエンド・ユーザーは高齢者とした．エンド・ユーザーには，研究のために事前に登録しておいたユーザーと，事前登録のない一般のユーザーの2種類がいる．登録ユーザーには，認証してシステムを利用させる．認証は高齢者が理解し使いやすいものとした．一般ユーザーは，公開された Web アプリにアクセスするさまざまな人を想定している．また，イベント等でプロジェクトが用意したタブレットなどを用いて回答してもらうこともあった．その場合，同一のデバイスを連続して異なる一般ユーザーが使用するが，それぞれ回答者が異なることを区別できるようにした．

3.3 アプリのフロー

アプリはプロジェクトで用意した一連の質問を一つずつ表示し，エンド・ユーザーはこれに回答していく．回答内容によって質問が変わることもある．回答していくとプロジェクトで用意したアルゴリズムにより診断を行い表示する．回答の途中でも中間の診断を表示することもある．また，回答途中でツールの使い方やサービスの提供元，問い合わせ先なども必要に応じて閲覧できるようにした．

3.4 表示

表示を行うプログラムは W3C の Web Content Accessibility Guidelines (WCAG) 2.1[3]，Accessible Rich Internet Applications (WAI-ARIA) 1.1[4]などのアクセシビリティの国際的ガイドラインに準拠し，高齢者が使いやすいものとした．また，高齢者が親しみやすく，見やすく，理解しやすく，操作しやすい表示を行うようにした．アクセシビリティだけでなく，画面から受ける印象に配慮した．たとえば，アプリや診断結果の理解を助けるデザインとし，高齢者に配慮した表示を行うものとした．

3.5 質問セット

アプリは，質問と回答と診断アルゴリズムと結果の表示のセット(質問セット)で構成されるものとした．

質問は，質問 ID，質問番号，質問の種類，本文から構成され，回答は択一式とした．質問は表示順序付きのセットとして扱い，複数の質問セットを切り替えて使用できるようにした．そして診断アルゴリズムを質問セットごとに設定できるようにした．アプリを研究でも使用するために，質問セットはエンド・ユーザーを指定して使用できるようにした．たとえば，サービスリリース当初は一般ユーザーには「質問セット A」を用い，数ヶ月運用して質問内容を改善し「質問セット B」に切り替える，あるいは指定した登録ユーザーには「質問セット C」を用いるなどの運用を可能とした．

診断アルゴリズムは，プロジェクトから提供するが，個々の回答内容に応じて設定されたスコアの値を集計するものであった．

結果は，診断アルゴリズムが算出した数値，それに基づいて表示するエンド・ユーザーの特性や詐欺脆弱性の概略や詳細，アドバイスなど複数の項目から構成され，その内容はプロジェクトから提供した．

3.6 データの記録

回答データとして，回答者を区別する情報，質問セット，質問 ID，回答内容，回答日時を関係データベースに記録した．回答データは，登録ユーザーだけでなく，個々の一般ユーザーも区別

して記録した．回答データは，履歴データであり，同一の回答者が同一の質問に複数回回答した場合にも，それぞれの回答を記録した．

また，診断結果のデータとして，回答者を区別する情報，質問セット，診断した日時，複数の項目から構成される診断内容を関係データベースに記録した．診断結果データも，履歴データであり，同一の回答者が同一の質問セットに複数回回答して複数回診断が行われた場合にも，それぞれの診断結果を記録した．

4. アプリの画面遷移

仕様に沿ってアプリの画面遷移の例を記述したのが図 1 である．図の角丸長方形は画面があるもの，長方形は画面がないものである．質問は回答していくと次々と表示される．質問の内容に応じて分岐したり，診断を表示することもある．分岐は表示画面がなく次の画面に遷移するだけである．なお，Web ではブラウザの「戻る」ボタンで逆方向に戻ることができるが，戻る方向の矢印は図 1 では省略している．

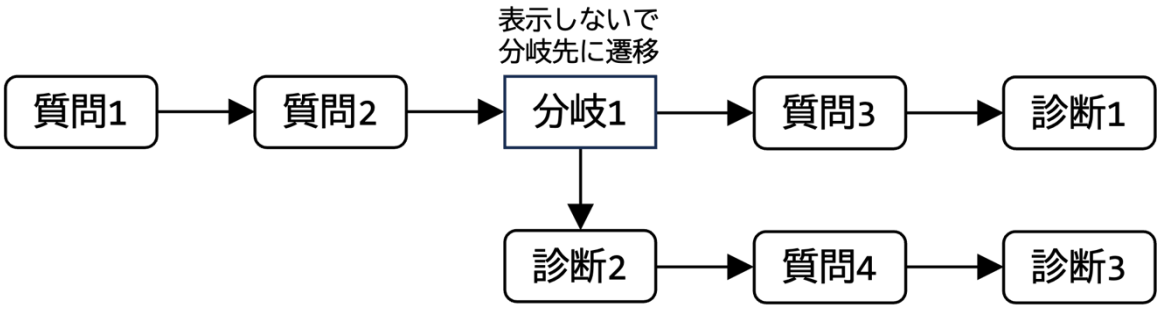

図 1 アプリの画面遷移の例

5. オブジェクト指向設計

Ruby on Rails では，MVC(Model-View-Controller)[5]の 3 種類のクラスを持ったアーキテクチャを採用している．モデル Model がデータを扱い，ビューView が表示を行い，コントローラ Controller が URL に対応したモデルとビューの操作を行う．アプリケーションでは，一般に表示は変更が頻繁に発生する可能性が高いことが知られている．そこで，表示をビューとして他の部分から分離して，変更の影響範囲を狭くしようとしたのが MVC アーキテクチャである．

ソフトウェアの変更は表示以外にも発生することがある．特に本アプリは実験的な性格を持ち，最終的に質問，診断アルゴリズム，表示をどうするのかを開発前に決めることができなかった．そのため，ある程度何かが途中で変わっても対応できるようにする必要があった．そこで，本アプリでは，変化に対応しやすいオブジェクト指向設計を積極的に取り入れた．具体的には，質問セットの「表示項目」は質問・分岐・診断など異なる種類のオブジェクトで構成されるが，同じ継承元を持たせ，多態性を用いて交換できるようにした．これにより，これらの項目を連結リストの要素として自由に繋げて構成できるようになった．また，診断アルゴリズムも，多態性を用いて簡単に交換することができるようにした．

5. アプリのドメインモデル

Ruby on Rails では，MVC の 3 種類のクラスに分けて開発を行う．このうち，筆者が開発したモデル Model とコントローラ Controller について論じる．

5.1 ドメインモデルの全体像

ドメインモデルとは，アプリが扱う対象のデータのクラスである．アプリ全体のクラス図を図 2 に示す．

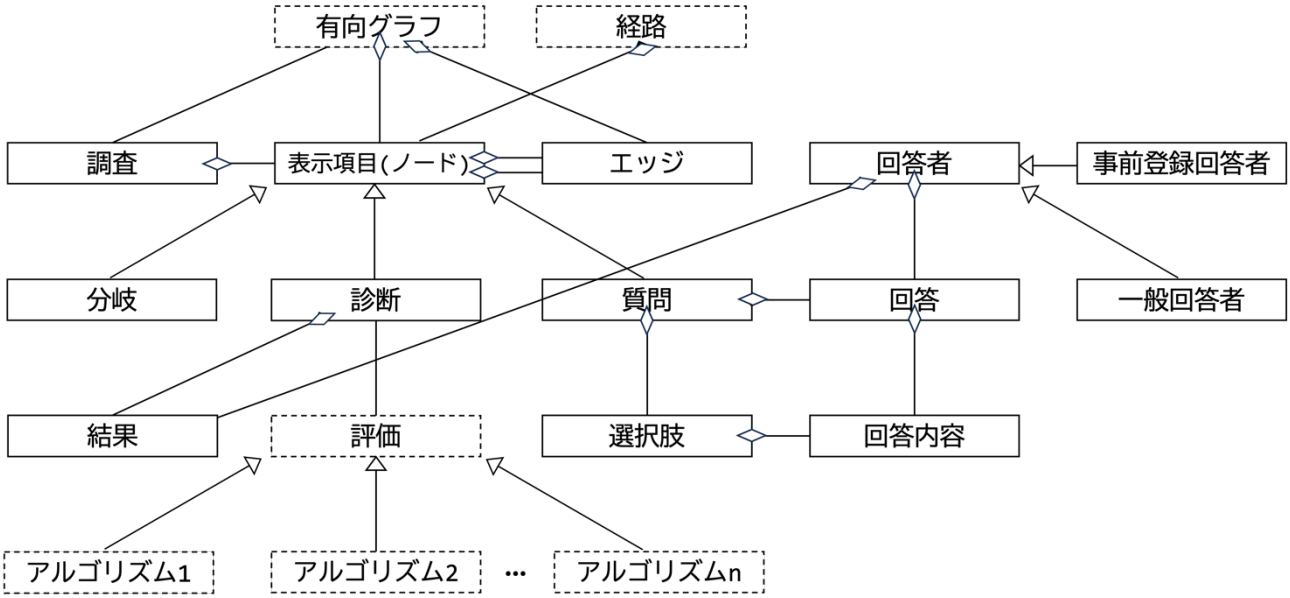

図 2 ドメインモデル全体のクラス図

図 2 のクラス図で長方形が個々のクラスを表す．長方形の中の文字がクラスの名前である．実線の長方形のクラスは，関係データベースに対応したテーブルが存在し，データは永続化されている．破線の長方形は，本論文独特の表記だが，データベースを使っていないところである．図 2 の上部の破線の有向グラフと経路は，グラフ理論を用いて回答状況の検証や回答の進捗度を計算するために使うものである．下部の評価やアルゴリズム 1，2 などの破線は診断アルゴリズムに関するものである．クラス間に引かれた線のうち先端に白い矢印のあるものは，クラスの継承関係を示し，矢印がある側が継承元，矢印がない方が継承先である．先端に菱形のあるものは，クラスの集約関係を示し，菱形がない方が菱形の方に複数所属することを示す．

以降では，部分に分けてモデルのクラスの詳細を論じる．

5.2 回答者に関するモデル

回答者に関するモデルのクラス図を図 3 に示す．図 2 ではクラスは長方形で表し，中にクラスの名前だけを記していたが，図 3 ではクラスの属性(2 行目)とメソッド(3 行目)を追加している．属性は「属性名:データ型」の形式で記載している．図 3 のクラスは，クラスごとに独自に用意したメソッドはない．煩雑になるため図では省略しているが，Ruby on Rails のモデルは ActiveRecord というクラスを継承している．ActiveRecord は Object-Relational マッパーの一種で，プログラムであるクラスとデータベースのテーブルをマッピングし，データベースの 1 行 1 行をクラスの 1 つ 1 つのインスタンスのように扱うことができるようにする．ActiveRecord を継承したクラスでは

ActiveRecord のメソッドを利用でき，データの CRUD(新規登録／読み出し／更新／削除)などをオブジェクトとして行うことができる．

図 3 では，「回答者」は整数 integer 型の id と文字列 string 型の type の 2 つの属性を持っている．研究用の「事前登録回答者」とそれ以外の「一般回答者」は，「回答者」クラスを継承している．「事前登録回答者」は認証して利用する．実際に実装した認証は，見間違いにくい英小文字と数字を選んで組み合わせた 4 桁×2 つの文字列を入力して行うものとした．これが「事前登録回答者」の属性「コード 1]「コード 2」である．高齢者にスマートフォンなどで英数を打ってもらうことは難しいと思われたので，QR コードも用意しカメラがあるデバイスでは文字を入力しなくても認証できるようにした．また，研究上「事前登録回答者」には「一般回答者」とは異なる調査を行うこともあるので，「調査の識別名」も属性として持たせている．

プログラムであるクラスには継承の機能があるので，クラス図及びプログラム上はこれで問題ない．しかし，関係データベースのテーブルには継承の機能がないため，このクラス図に沿ってテーブルを作ることができないインピーダンス・ミスマッチと呼ばれる問題が発生する．この問題には，シングル・テーブル継承 [6]という現実的な解決方法が知られている．それは関係するクラスのすべての属性を持つテーブルを 1 つ用意し，関係するクラスをその 1 つのテーブルの部分列にマッピングすることである．そして，type という名前の属性を用意し，type には対応するクラスの名前を記録する．つまり，この場合，表 1 のように「回答者」という親クラスに対応したテーブルを用意する．テーブルには id，type，コード 1，コード 2，調査の識別名というすべてのカラムを持たせる．type には「事前登録回答者」「一般回答者」などの文字列が記録される．たとえば，type の値が「事前登録回答者」の場合，そのレコードは「事前登録回答者」のインスタンスにマッピングする．「一般回答者」に対応したレコードでは「コード 1]「コード 2」「調査の識別名」は使用されず，実際には NULL になる．

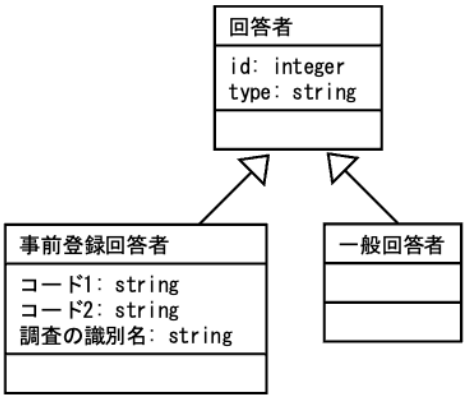

図 3 回答者に関するモデルのクラス図

表 1 シングル・テーブル継承の「回答者」テーブル

id	type	コード 1	コード 2	調査の識別名
1	事前登録回答者	h36x	ue8w	研究用 1
2	一般回答者	NULL	NULL	NULL
3	事前登録回答者	1dv4	xm2h	研究用 2
⋮	⋮	⋮	⋮	⋮

5.3 表示項目に関するモデル

図4に表示項目に関するモデルのクラス図を示した．このアプリでは，回答中に画面に表示されるのは「質問」と「診断」である．これをまとめる「表示項目」という親クラスを用意した．また，それまでの回答内容によっては異なる経路を取る場合があるので，画面には表示されないが「分岐」というクラスも用意し，「分岐」も「表示項目」を継承している．これにより，「質問」「診断」「分岐」は互いに交換することができるようになった．「表示項目」の継承にも「回答者」と同様にシングル・テーブル継承を用いていて，「表示項目」は「type」属性を持っている．また，「表示項目」を連結リストとして扱えるように「次の表示項目」メソッドを持たせている．これらにより，「表示項目」を自由に組み合わせて診断を作れるようになり，自由度が高まった．

それから，ある「調査」には一連の「表示項目」が含まれるものとし，図4のクラス図では「調査」は「表示項目」を集約している．集約方法はいくつかあるが，データベースのテーブルは正規化されている必要がある．正規化するために「表示項目」に，集約元の「調査 id」を属性として持たせた．その他に「表示項目」は「id」「識別名」の属性を持つ．「識別名」には「問 1」などの値が入り，画面のタイトルに使われる．

「質問」には「文」「入力の種類」の属性を用意した．「入力の種類」は HTML の input 要素の type 属性の値で，択一選択「radio」，複数選択「checkbox」などの値を持つ．また，次の質問を指すのに「次の表示項目の識別名」属性を用意し，その値は「次の表示項目」メソッドで使用される．

「分岐」「診断」クラスは，これまでの回答に何らかの評価を行うものである．「評価の実行」メソッドには，これまでの「回答のハッシュ」を渡すと，「評価クラス名」の評価を実行する．「分岐」の場合は，何も表示せずに「次の表示項目」メソッドの結果に従って「次の表示項目」に遷移する．「診断」は診断結果を表示する．

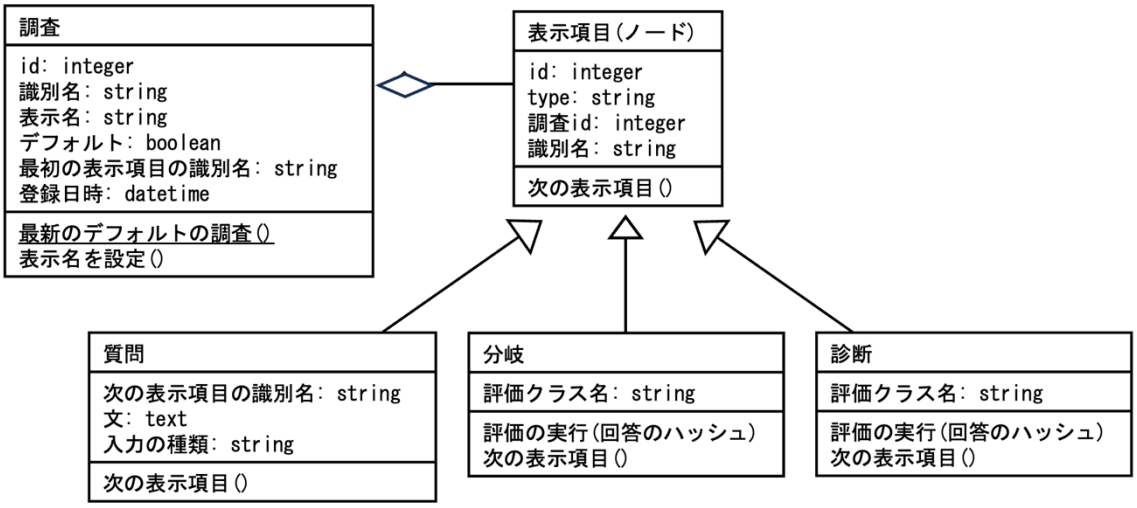

図 4 表示項目に関するモデルのクラス図

「調査」には「id」「識別名」「表示名」を持たせた．調査は細かいバージョンも含めて記録したいので，表示上は同じ名前でも，バージョンが異なることがあり，それぞれ「表示名」「識別名」とした．「表示名」は同じことがあるが，「識別名」にはバージョンがわかる名前を入れ，同一のものはないようにする．また，「一般回答者」に表示する「調査」を指定するために，「デフォルト」「登録日時」を属性として持たせた．「デフォルト」が true で，「登録日時」が最新の調査を「一般回答者」に表示する仕様とした．

5.4 有向グラフに関するモデル

公開初版のアプリには，回答が記録されないことがあるという不具合があった．また，回答の進捗度を表示したいという要望があった．そこで，グラフ理論を用いたデータ構造を利用することにして図 5 のクラスを用意した．

グラフは，ノードという点とエッジというノードとノードを繋ぐ線で構成される．アプリの場合，表示項目がノードに相当するので，そのままノードとして使用する．エッジはノードである表示項目をデータベースに登録するときに，「次の表示項目」メソッドの結果を利用して，一緒にデータベースに登録する．エッジは「id」「始ノード id」「終ノード id」を属性として持つ．「始ノード id」「終ノード id」は，それぞれ表示項目の「id」である．また「==」というメソッドを用意し，エッジ同士を比較して同一のエッジか否かを調べられるようにした．

「有向グラフ」や現在の回答の「経路」は，その他のモデルのデータや回答状況を使うと導出することができるので，データベースに保存せず，回答時にメモリ上に生成することにした．データベースのテーブルを使わないので，テーブルを正規化する必要はなく，集約元に集約する対象のクラスの情報を持たせた．具体的には「有向グラフ」は「調査」とノードの集合「ノード群」とエッジの集合「エッジ群」を持つ．経路はノードの集合の「ノード群」を持つ．

「有向グラフ」は，調査ごとに作られる．Web では URL を指定すると任意のページにアクセスできるので，ある調査を表示しているときに，別の調査の質問などにアクセスすることも可能になるが，これは調査上問題である．アクセスした先が回答中の調査のものであることを保証するために，ノードやエッジが有向グラフに含まれているかを確認するメソッド「含むか」を用意して回答中に検証を行うようにした．

それから，回答中に進捗状況を表示したいという要望が寄せられた．進捗を求めるには，現在のノードから終端ノードまでの最長経路の長さを，最初のノードから終端ノードまでの最長経路の長さで割ればいい．長さはこの場合，経路に含まれるノード数である．経路は分岐する可能性があるので，終端ノードは複数あり得るし，それぞれの終端ノードに到達する経路も複数ありえる．そこで，すべての経路を調べるために，あるノードの「子ノード群」を列挙するメソッドを用意し，これを繰り返し再帰的に使用した．そうして求めた「終端ノード群」を列挙するメソッドを用意した．これにより，あるノードからすべての終端ノードまでの距離を求めて，「終端ノードへの最長経路」を求めるメソッドを用意した．また，デバッグ用に有向グラフを「文字列化」して表示するメソッドを用意した．

「経路」には，回答経路が途切れていないかを検証して，通過したエッジを「追加」していく．ただし，属性としてはエッジ群(エッジ A→B，エッジ B→C，…)よりもシンプルなノード群(ノード A，ノード B，ノード C，…)を持たせた．また，Web で回答していると，ブラウザの「戻る」ボタンで前のページ戻ることがある．この動作は JavaScript を用いると抑制することができるが，ブラウザのデフォルトの動作を変更するため，アクセシビリティ上は望ましくない．そこでブラウザの「戻る」ボタンでページを戻ることを抑制しないことにした．なお，サーバーサイドにノードの移動が記録されるのは進んだ時だけで，ブラウザの「戻る」ボタンで戻った分は記録されない．そこで再び回答を進めたときに回答経路が切れていないことを検証するために，進んだ時に追加しようとするエッジの始ノードが回答済の「ノード群」に「含むか」どうかを調べることにして，そのメソッドを用意した．

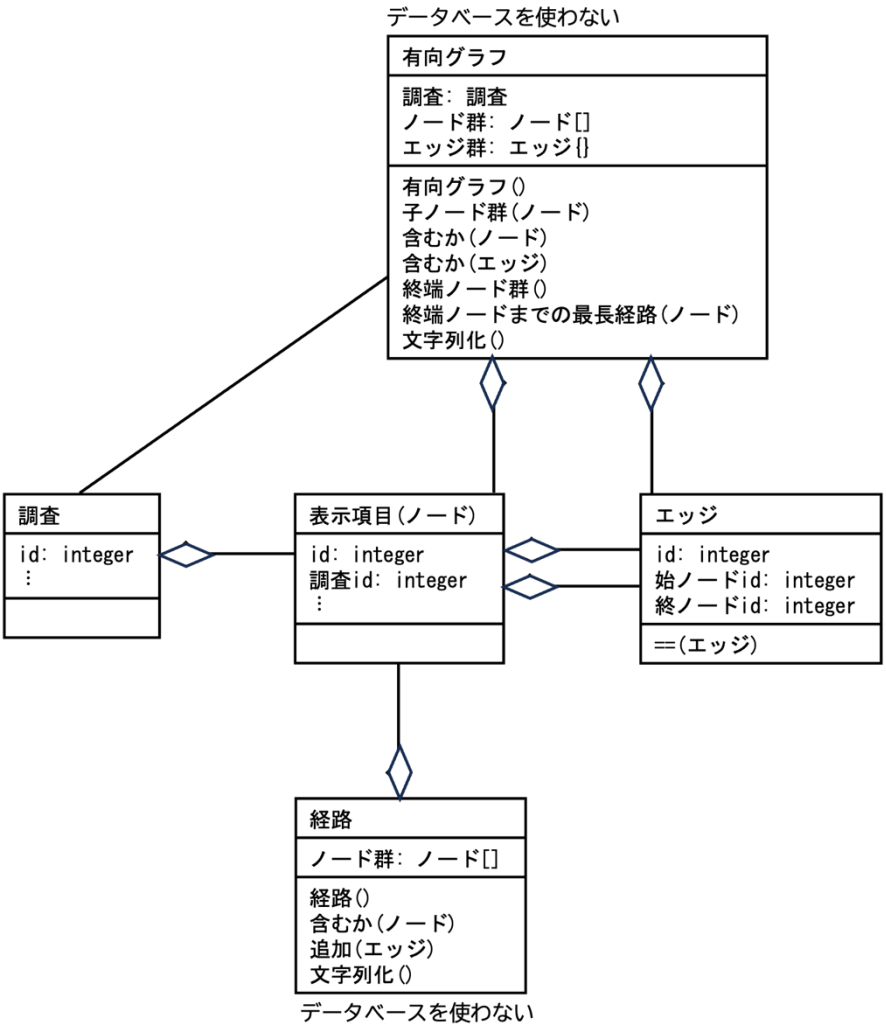

図 5 有向グラフに関するモデルのクラス図

5.5 質問と回答に関するモデル

質問と回答に関するモデルのクラス図を図 6 に示した.「質問」は「表示項目」を継承していて,これらの説明は「表示項目」に関するモデルの説明で行なっている.「質問」には複数の「選択肢」があり,これらは集約関係にあり,「選択肢」は「表示項目 id」を属性に持つ.「選択肢」には画面に表示される「文」とそれを選択したときにサーバーに送られる「値」を属性に持つ.「選択肢」の表示順は「登録日時」属性の順になる.

「質問」には「回答」がある.「回答」は「質問」と「回答者」と集約関係にあり,対応して「表示項目 id」と「回答者 id」を属性として持つ.Web アプリなので,回答中に「質問」を前後して回答し直し,同じ「質問」に複数回回答する可能性がある.最終的な回答を特定するために「回答日時」を属性として持つ.

回答が択一選択の場合,「回答」に選択した「選択肢」の情報を持たせることもできるが,複数選択を可能にすると,その値が複数個になる可能性がある.すると,テーブルを正規化して分割する必要がある.そこで「回答」とその「回答内容」に分割した.「回答内容」は「選択肢 id」と「回答 id」を属性として持つ.

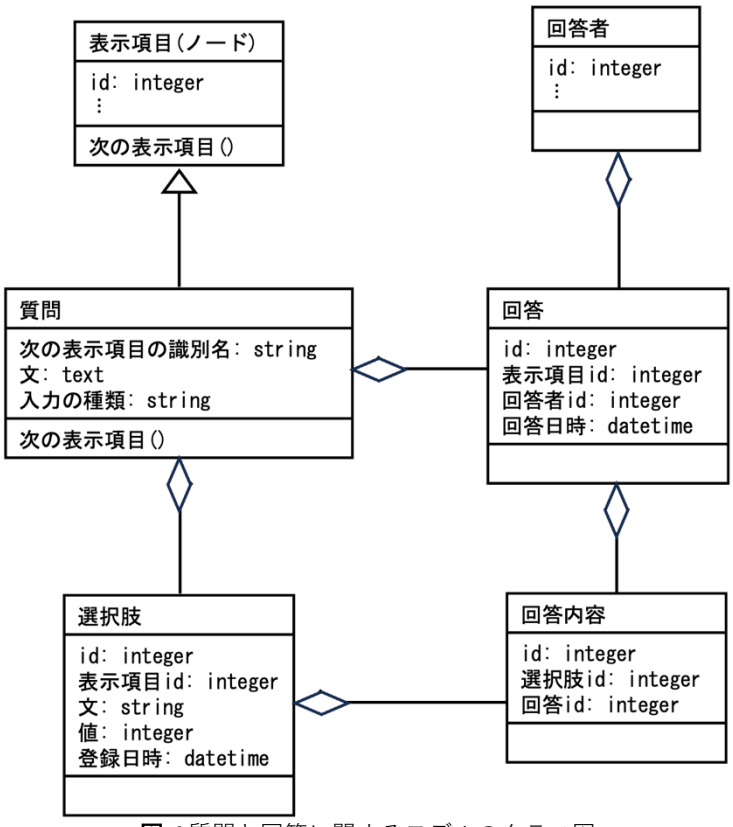

図 6 質問と回答に関するモデルのクラス図

5.6 診断に関するモデル

診断に関するモデルのクラス図を図 7 に示した.「診断」は「表示項目」を継承していて，これらの説明は「表示項目」に関するモデルの説明で行なっている.「診断」には複数の「結果」があり，これらは集約関係にあり,「結果」は「表示項目 id」を属性に持つ. また,「結果」は属性として「回答者 id」, 診断に使った「回答一覧 JSON」と結果の「概要 JSON」「詳細 JSON」を持つ. JSON は JavaScript Object Notation のことで，構造化されたデータのハッシュを文字列化したものである. JSON 形式のデータは Web におけるデータ交換でよく用いられ，さまざまなプログラミング言語でサポートされている.

「診断」には「評価」を用いる. 研究にも使うアプリなので,「評価」にはさまざまなアルゴリズムがあり得る. 急に新しいアルゴリズムが必要になっても対応できるように，それらの共通部分を「評価」とし，評価に使うクラス名を属性としてもち，それまでの回答をハッシュとして引数に渡して評価を「実行」するメソッドを持たせた. それぞれのアルゴリズムはこれを継承するだけで使用することができる. アルゴリズムのプログラムはクラスのファイルに記述できるものであれば何でも使用できるようになった. 実際に使用されたのは，ロジスティック回帰で，そのパラメータを CSV ファイルから読み込んで使用していた.

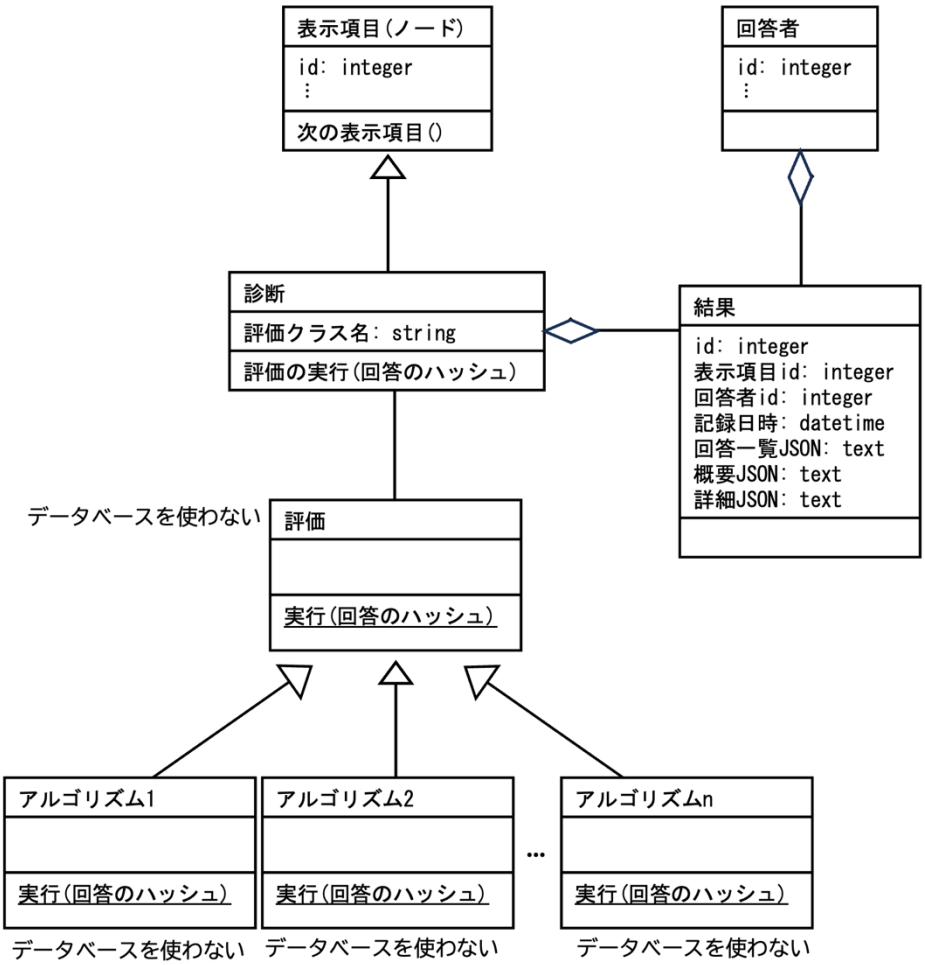

図 7 診断に関するモデルのクラス図

6. アプリのコントローラ，ヘルパーのクラス

モデルに引き続き，アプリのコントローラのクラスを論じる．また，コントローラで表示や認証の補助に繰り返し使う小さいプログラムをヘルパーというが，それについても論じる．図 8 にコントローラとヘルパーの全体像を示した．メソッドの先頭に「-」がついているものは，そのクラス自身あるいは継承先のクラスの内部でだけ使用するプライベート・メソッド，「+」がついているものは外部から呼び出せるパブリック・メソッドである．

モデルの多くがデータベースをマッピングする ActiveRecord を継承しているように，コントローラは「アプリケーション・コントローラ」を継承している．モデルの場合，ActiveRecord の継承を記述すると大変煩雑になる．また，ActiveRecord に追加のコードを記述せずに使用していたので，記載は省略した．一方「アプリケーション・コントローラ」には「ログインの確認」「ログアウトの確認」「回答の進捗度」の 3 つメソッドを追加している．これは多くのコントローラで共通して使うメソッドである．また，ヘルパー・メソッドを追加した「アプリケーション・ヘルパー」「セッション・ヘルパー」を読み込んでいる．

「アプリケーション・ヘルパー」にはタイトルを表示するための「完全なタイトル」メソッドを記述している．「セッションヘルパー」には認証と認可とセッションをサポートする「ログイン」「現在の回答者か？」「現在の回答者」「ログインしているか？」「ログアウト」などのヘルパー・

メソッドを記述している．

すべてのコントローラーは「アプリケーション・コントローラ」を継承しているが，「質問コントローラ」「回答コントローラ」「分岐コントローラ」は共通部分が多いので，「表示項目コントローラ」を親クラスとして用意し，それを継承するようにした．「調査コントローラ」も表示項目を表示するコントローラの一種だが，回答者に割り当てられた「調査を表示」したり，デバッグ用に「調査一覧ページ」を表示したりするなど，「表示項目コントローラ」と差異が多いので，「表示項目コントローラ」を継承させていない．診断の表示も他の表示項目と差異が大きいので「診断コントローラ」も同様に継承させていない．

「静的ページ・コントローラ」は質問の回答と関係ない，「トップ・ページ」「ツールについてページ」「お問い合わせページ」「ヘルプ・ページ」「プライバシー・ポリシー・ページ」など，固定の内容を表示する．

「セッション・コントローラ」は，ログインやログアウトなどの処理を行うメソッドを記述した．

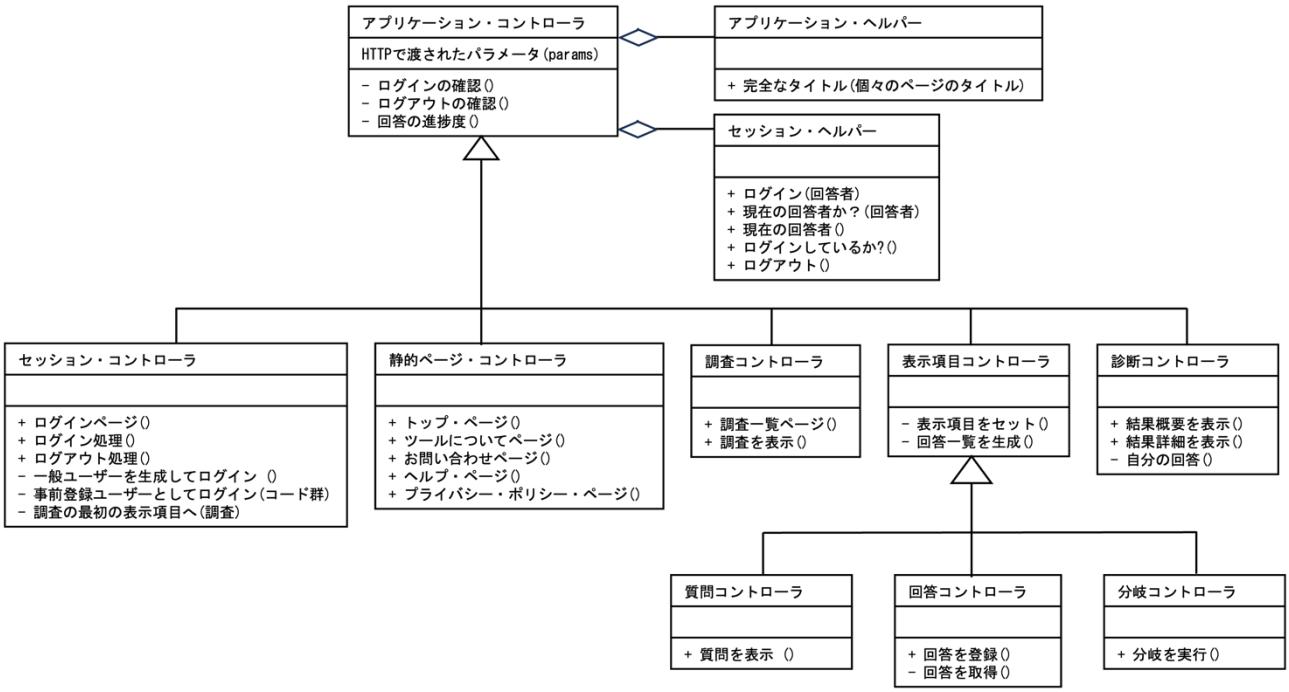

図 8 コントローラ，ヘルパーのクラス図

7. まとめ

本論文では，詐欺抵抗力診断アプリのオブジェクト指向設計について論じた．このアプリは，研究用にも用いられたが，研究では先の見通しが必ずしも立っているわけではなく，さまざまな変更が予測された．そこで，オブジェクト指向設計を活用し，柔軟で変化に強い設計を行った．具体的には，オブジェクトの継承を利用した多態性を用いて，画面に表示する質問や診断を分岐と組み合わせて自由に組み立てられるようにした．また，一般の回答者と認証して利用する研究用の事前登録回答者，さまざまな診断アルゴリズムにも多態性を用いて，自由度を高めた．一方，オブジェクトをマッピングする関係データベースのテーブルには継承の機能がないため，プログラムとデータベースの間でインピーダンス・ミスマッチが発生する．このミスマッチを解消するためにシングル・テーブル継承を用いて，データベースのテーブルでの継承を実現した．

設計に沿って Ruby on Rails で実装してみると，診断のバージョンごとにビューが増え，これを効率的にわかりやすく管理する方法が提供されていないことがわかった．また同様に，診断のアルゴリズムも複数登録して使用することができるが，これも増加する一方で効率的にわかりやすく管理することは難しくなる傾向があった．

謝辞

本研究は，国立研究開発法人科学技術振興機構(JST)の社会技術研究開発センター(RISTEX)の戦略的創造研究推進事業(社会技術研究開発)「安全な暮らしをつくる新しい公／私空間の構築」研究開発領域に平成 29 年度に採択された「高齢者の詐欺被害を防ぐしなやかな地域連携モデルの研究開発」(研究代表者・渡部諭・元秋田県立大学教授)のものである．詐欺抵抗力診断アプリ「わたなべ教授のサギ抵抗力しんだ〜ん」に回答くださったみなさん，JST，研究開発領域の領域総括およびアドバイザーのみなさん，プロジェクトのメンバーみなさんに感謝いたします．

参考文献

1) 科学技術振興機構 社会技術研究開発センター. 高齢者の詐欺被害を防ぐしなやかな地域連携モデルの研究開発. https://www.jst.go.jp/ristex/pp/project/h29_5.html <2023 年 12 月 1 日アクセス>
2) 渡部 諭, 岩田 美奈子, 上野 大介, 江口 洋子, 小久保 温, 澁谷 泰秀, 大工 泰裕. & 藤田 卓仙. (2018). 高齢者の詐欺被害を防ぐしなやかな地域連携モデルの研究開発. 秋田県立大学ウェブジャーナル A (地域貢献部門), 5, 64-72. http://id.nii.ac.jp/1180/00000765/ <2023 年 12 月 1 日アクセス>
3) Kirkpatrick, A., Connor, J. O., Campbell, A., & Cooper, M. (2018). Web content accessibility guidelines (WCAG) 2.1. WWW Consortium (W3C).
4) Diggs, D., McCarron, S., Cooper, M., Schwerdtfeger, R., & Craig, J. (2017). Accessible Rich Internet Applications (WAI-ARIA) 1.1. WWW Consortium (W3C).
5) Reenskaug, T. (1979). Mvc xerox parc 1978-79. Trygve/MVC.
6) Fowler, M., 長瀬嘉秀, 株式会社テクノロジックアート. (2005). エンタープライズアプリケーションアーキテクチャパターン. 翔泳社.

要旨

この論文では，詐欺抵抗力診断 Web アプリ「わたなべ教授のサギ抵抗力しんだ〜ん」のオブジェクト指向設計について論じる．

このアプリは Web で一般公開されたが，研究にも利用された．研究の場合，開始時に最終的な完成状態を予測することは難しい．開発や運用している途中で，たくさん変更が発生することが予測された．そこで，オブジェクト指向設計を活用し，変化に強いアプリを開発した．アプリでは，質問と分岐と診断を自由に組み合わせて複数の調査を作ることができる．また，診断に用いるアルゴリズムも任意のものを用いることができる．

キーワード: オブジェクト指向設計, 高い自由度, 研究用, Web アプリケーション