Android 診断アプリのコンテンツ改修の容易性に関する検討:「南部昔ッコ診断」の事例

小久保 温[†]・小玉 成人[†]・伊藤 智也[††]

A Study of Modifiability of Android Application's Content: A Case of "Personality Test of Characters Types of Folk Tales from Nanbu"

Atsushi KOKUBO[†], Naruhito KODAMA[†] and Tomoya ITO[††]

ABSTRACT

We developed the "Personality Test of Characters Types of Folk Tales from Nanbu" app (henceforth called "Nanbu" app) by replacing the contents of the "Personality Test of Hachinohe Pioneers Types" app (henceforth called "Hachinohe" app). The first time we developed "Hachinohe" app, it took about two months. However, the development of the "Nanbu" app took only two days in fact. This was because the application was divided into data and program, and the program was further divided into MVC to narrow the scope of influence when changes occurred. The task of replacing the content was mainly to create the graphic design. Ideally, when replacing content, it would be possible to replace it by changing only the data. However, it is necessary to change the view, which is the program that manages the screen of the application. We will introduce the knowledge we learned from this content renovation to make it easier to change the view program.

Key Words: modifiability of software, MVC, Android appplication

キーワード : ソフトウェアの改修の容易性, MVC, Androidアプリ

1. はじめに

「南部昔ッコ診断」は，八戸市博物館と八戸工業大学の共同研究によりProcessing[1)]で開発したAndroidアプリケーションである。「昔ッコ」とは，昔話のことである。八戸市博物館令和2年秋季特別展「暮らしの中の手仕事」に出展する予定であったが，開発担当者の小久保が開発を予定していた期間に事故により体調を崩したため，令和2年度末の納品となり，令和3年5月中旬に八戸市博物館で展示された。本アプリケーションは，令和元年に開発した「はちのへ先人診断」のコンテンツを入れ替えたもので，開発に要した期間は2日間であった。本報告では，コンテンツの改修を通じて得られた知見を検討する。

2.「はちのへ先人診断」

「はちのへ先人診断」は，2019年(令和元年)に

令和 3 年 12 月 6 日 受付

† 工学部システム情報工学科／大学院工学研究科電子電気・情報工学専攻・教授

†† 工学部システム情報工学科／大学院工学研究科電子電気・情報工学専攻・准教授

火発し，八戸市博物館で2019年7月13日(土)から8月25日(日)まで市政施行90周年記念特別展「八戸90年の歩み」で展示された。

開発のスケジュールは，2018年9月26日に企画をうちわせ，2019年4月18日に具体的な内容に関して相談し，5月8日にアプリの素案とデータ，6月18日にキャラクターの写真を受け取り，7月6日に納品している。開発期間は約2ヶ月であった。

開発内容については「Androidアプリ「はちのへ先人診断」の開発」という同名のタイトルで，八戸工業大学[2]と八戸市博物館[3]の紀要に論文を投稿した。タイトルは全く同じだが，八戸工業大学のものは技術的内容で，八戸市博物館のものは展覧会展示物に関する内容で異っている。

「はちのへ先人診断」は，八戸の先人に関係した10問の質問に回答していくと，線形分類器により，どの先人に似ているかを診断する診断アプリである。診断アプリとは，ユーザーがどういう人なのかを診断するアプリの総称で，Web上では人気を博している。

「はちのへ先人診断」の画面遷移は図1のようになっている。診断結果は概要が表示される「判定」画面と詳しい説明が出る「詳細」に分かれている。

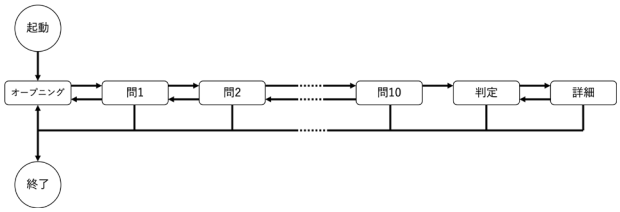

図1「はちのへ先人診断」の画面遷移

3. アプリの設計と改修の容易性

大きなシステムは複数のメンバーが分担して作るものである。分担しやすくするには，部分に分ける必要があり，それぞれの部分の変更が他の部分の変更を要求しないように，部分間の結合が疎になるように設計することがポイントである。システムには変更が発生するものなので，仮に1人で開発する場合にも，変更箇所を狭い範囲に限定することは，開発を容易にするために有用な方針である。

たとえば，アプリはプログラムとデータから構成される。このとき，プログラムを変更するにはプログラマが必要だが，データは必ずしもそうではない。よって，プログラムとデータを分け，それぞれの仕様を共有することで分担して開発しやすくなる。

さらに，プログラム自体もより細かく分割することで開発しやすくなる可能性がある。プログラムを分割する上で参考になるものの一つが，MVCアーキテクチャ[4]である。MVCは，プログラムをデータを扱うモデル(Model)，表示を行うビュー(View)，ユーザーの入力を受け取りモデルとビューを操作するコントラーラー(Contoller)に分ける方法である。このうち，表示を変更したいという要望はよく発生するため，ビューにはしばしば変更が発生する。しかし，データの構造に変更がなければモデルに変更は発生しない。MVCに分けることで，変更を狭い範囲にとどめて開発を容易にすることができる。なお，モデルの構造を変更すると表示が変わるので，ビューとコントローラーの変更は必然的に必要となり，変更箇所が多岐に渡ってしまう。よって，データの構造を変更するのには慎重になる必要がある。

4. アプリの構造

改修の容易性という観点から「はちのへ先人診断」は，データとプログラムを分け，さらにプログラムをMVCに分割して開発した。

アプリのデータにはCSVファイルと画像データの2種類がある。CSVファイルは，Microsoft Excelで作成し，「CSV UTF-8(コンマ区切り)」形式で保存したものである。UTF-8で保存しているのは，開発に使用したProcessingのデフォルトがUTF-8だからである。CSVファイルには出題される質問や選択肢，先人のデータ，線形分類器で診断する

ための各質問と各先人の組み合わせに割り振られた係数の3種類がある。画像はMicrosoft PowerPointで作成し，PNG形式で出力したものを使用している。

プログラムはMVCに分け，モデルはCSVのデータファイルから質問のデータを読み込み，連結リストからなる質問と質問が持っている選択肢を構成する。ビューは質問や結果などの画面の種類と内容に応じて画像を読み込み，画面(シーン)を作り，前の画面に戻れるようにスタックで管理する。コントローラーはボタンをタップしたときの振舞いとして表現する。

5. コンテンツを改修するために発生した作業

アプリを「はちのへ先人診断」から「南部昔ッコ診断」に改修するために要した期間は，実質2021年3月23，24日の2日間であった。この作業を次にまとめる。

5.1 八戸市博物館から受け取ったデータ

八戸市博物館から，改修するために受け取ったのは，アプリの文言，質問と選択肢(Excel形式)，診断結果に出るキャラクターの画像，線形分類器で診断するために使用する各質問とキャラクターの組み合わせの係数の4つで，これはアプリのデータであった。また，「はちのへ先人診断」で使用しているデータと同じフォーマットであり，中を確認しただけで変更せずに使用できることがわかった。

5.2 グラフィックデザインの制作

提供されたデータには，画面のデザインは含まれていなかった。そこで，最初に画面のグラフィックデザインをMicrosoft PowerPointで制作した(図2〜5)。これに開発期間の2日の半分以上を費やした。

まず，表示するデータの構造が「はちのへ先人診断」と共通だったため，レイアウトは「はちのへ先人診断」を踏襲した。ただし，表示さ

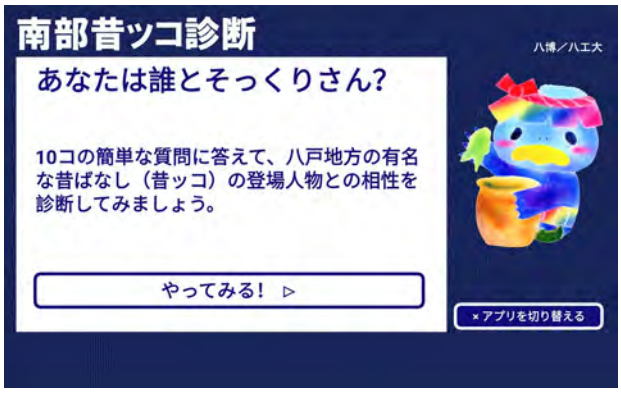

図2「はちのへ先人診断」の画面遷移

図3「はちのへ先人診断」の画面遷移

図4「はちのへ先人診断」の画面遷移

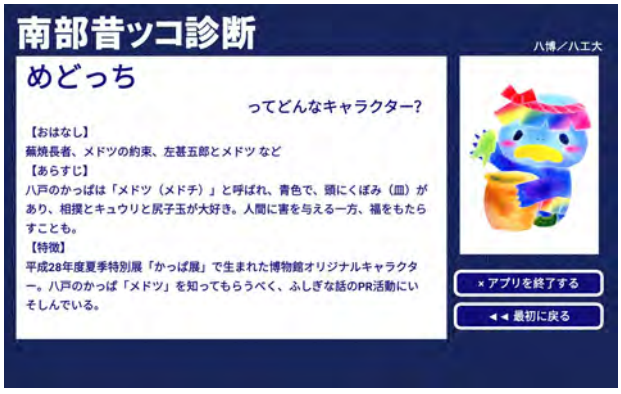

図5「はちのへ先人診断」の画面遷移

れる文字数が異なるので，文字のサイズを調整している。時間がかかったのは，南部らしく，かつはっきり見えるコントラスト比の高い配色を選ぶことだった。

5.3 ビューのプログラムの改修

グラフィックデザインの変更は，ビューのプログラムにも影響する。ビューでは，プログラムで座標と色を指定して動的に文章やボタンを作り出して表示している。そのため，文字やボタンや画像の配色や配置した座標が変わったりすると，その部分を変更する必要がある。

パーツの配色や座標は，デザインを作成したときに使ったPowerPointで値を調べることができる。配色は，macOS版のPowerPointバージョン16では，図6のように表示される。RGBで設定するモードでは，レッド，グリーン，ブルーの値(0～255)と16進数カラー値(000000～FFFFFF)のどちらでも色の値の表示と設定ができる。この値を記録しておけば，Processingで使うことができる。また，不透明度(0～100%)も同様である。ただし，Processingの場合は不透明度の値の範囲は0～255であるため注意が必要である。

サイズと位置は，macOS版のPowerPointバージョン16では図7のように表示される。こちらはそのままProcessingで使うことはできない。まず，PowerPointでは表示される単位がcmであり，これをProcessingの単位であるピクセルに換算する必要がある。この換算は，macOS版PowerPointバージョン16では，画面の解像度が72dpiと見なされ，1インチが2.54cmなので，72÷2.54=28.3となり，cm単位の値を28.3倍することで計算できる。なお，エクスポートする画像サイズを指定でき，想定した画像サイズで出力する必要がある。しかし，OSやPowerPointのバージョンによっては，この換算が異なったり，出力画像サイズが指定できないことがあるので注意が必要である。たとえば，Windows版のバージョン16ではスライドのサイズをpx(ピクセル)単位で指定すると120dpiに換算されたcmになり，これをエクスポートするときは96dpiに換算された画像が出力される。

図6 macOS版Microsoft PowerPointバージョン16でのオブジェクトの配色の設定ダイアログ

図7 macOS版Microsoft PowerPointバージョン16でのオブジェクトのサイズと位置の設定ダイアログ

また，PowerPointではサイズは高さ，幅の順番で，位置は横位置，縦位置の順番で表示されており，サイズと位置で横縦の順番が逆である。一方，Processingでは，横方法，縦方向の順番で描画関数の引数を指定するように統一されている。このため，PowerPointで調べた値を，Processingで指定するには，換算と順番の入れ替えが必要で，手間がかかり，作業ミスも発生しやすい。著者は，図8のようにオブジェクトのサイズと位置をピクセル単位で記録したスライドも作ってから作業するようにしているが，その分作業が増える。

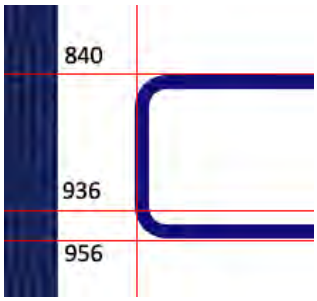

図8 ピクセル単位の座標を記入したスライド

具体的なプログラムは，次のようになっている。Processingは，Javaをベースにしており，Javaのプログラムの多くをそのまま使うことができる。まず，配色を使用する箇所で毎回指定すると矛盾した値を指定する危険性があり，変更した場合に修正箇所が増える。これを避けるために一元管理した方がよいので，最初に変数に値を指定しておく。ここではcolor()関数を用いて色を生成している。そして，final修飾子を使用して，これ以降は値を変更できなくした変数を用いている。

```
final color INDIGO = color(0, 32, 96);
final color LIGHT_GRAY
  = color(217, 217, 217);
final color YELLOW_GREEN
  = color(146, 208, 80);
final color DARK_GREEN
  = color(89, 128, 49);
```

配色と位置とサイズをボタンに設定するには，次のボタンのコンストラクタを使用して，ボタンを生成したときに設定するようにしている。コンストラクタでは，ボタンの配置とサイズ，ノーマル状態とアクティブ状態の文字，背景，枠線の色を指定する。

```
Button(表示する文字列,
  文字列を表示するフォント,
  x座標，y座標，幅，高さ，角丸の半径,
  ノーマル状態の文字色,
  アクティブ状態の文字色,
  ノーマル状態の背景色,
  アクティブ状態の背景色,
  枠線の太さ,
  ノーマル状態の枠線の色,
  アクティブ状態の枠線の色,
  サウンドエフェクト,
  振舞い)
```

これを具体的に記述すると，次のようになる。引数の個数が多く，どの引数が何を表しているのかがわかりにくく，修正しにくいものであった。

```
Button startButton = new Button(
  "やってみる！ ▷",
  largeButtonFont,
  265, 870, 850, 115, 20,
  INDIGO, LIGHT_GRAY,
  YELLOW_GREEN, DARK_GREEN,
  0, YELLOW_GREEN, DARK_GREEN,
  nextSoundEffect,
  new NextBehavior());
```

6. 改修を容易にする方法の考察

今回，改修するのに要した日数は実質2日間であった。しかし，変更するべき箇所が多く，わかりにくく，より効率化することができる可能性が示唆された。

たとえば，手間がかかり，ミスが発生しやすかったものとして，前節で取り上げたボタンオブジェクトの生成がある。これを改善してボタンのコンストラクタで指定している配色，位置とサイズ(と枠線の太さ)をそれぞれオブジェクトにすると，次のように大幅にコンストラクタを単純化することができる。

```
Button(表示する文字列，
  文字列を表示するフォント
  位置とサイズ，
  配色，
  サウンドエフェクト，
  振舞い)
```

元々のボタンのコンストラクタでは，16個の引数を指定していたが，個数が多くどの値が何を表すのかがわかりにくく，ミスを招きやすく，改修が困難であった。Pythonなどでは関数で名前付き引数が使えるため，1番目の引数に「″やってみる！▷″」を指定するのではなく，任意の順番で「statement: ″やってみる！▷″」と指定することができる。ここで「statement」は，ボタンに表示する文字列を表す引数に付けた名前である。名前付き引数が使えると，16個の引数があるコンストラクタでも，何番目の引数が何を表すのかを記憶してプログラミングする必要はなくなる。しかし，ProcessingがベースにしているJavaでは，名前付き引数は使えないので，引数の個数を減らしてわかりやすくすることが有効である。この新しいコンストラクタでは，6個の値を指定するだけでよい。

新しいコンストラクタを使用するには，次のように位置とサイズ，配色を表すクラスを作ればよい。配色の方は，ノーマル状態とアクティブ状態の配色を持った配色スキームクラス，それぞれの状態の文字，背景，枠線の色を配色クラスに分割すると，個々のコンストラクタの引数が更に減り，わかりやすく，ミスをしにくくなると思われる。

```
class 位置とサイズ {
  float x座標，y座標;
  float 幅，高さ;
  float 角丸の半径，枠線の太さ;
}

class 配色スキーム {
  配色 ノーマル状態;
  配色 アクティブ状態;
}

class 配色 {
  color 文字色;
  color 背景色;
  color 枠線の色;
}
```

また，長さの換算を別途紙上などで行って，換算後の値をプログラムに直接記述すると，プログラムを見ても何を意味しているのかわかりにくくなる。そこで，PowerPointで使用されているcmを，ピクセルに変換する関数を用意するとわかりやすく，変更しやすいプログラムになると思われる。macOS版のPowerPointバージョン16で調べた長さを，ピクセルに換算する関数は次のようになる。

```
int px(float cm) {
  return int(cm * 28.3);
}
```

この関数を使うと，ボタンのコンストラクタでx，y座標と幅と高さと角丸の半径を指定していた箇所は，次のようにPowerPointで表示された値

をそのまま記述することができる。また，PowerPointの仕様が変わった場合にも，関数で「*28.3」を行なっている箇所を1箇所変更するだけでよい。

```
265, 870, 850, 115, 20
```

↓

```
px(9.36), px(10.3), px(10.1), px(4.06),
px(0.71)
```

7. まとめ

「南部昔ッコ診断」は「はちのへ先人診断」のコンテンツを入れ替えたアプリケーションである。「はちのへ先人診断」の開発にはおよそ2ヶ月かかったが，「南部昔ッコ診断」はおよそ2日間であった。

改修が短期間で可能となったのは，改修の際の変更箇所が少なく狭い範囲に限定されるように，プログラムにデータを埋め込まず分離したこと，プログラムをMVCに分割しことが有効に働いたと思われる。MVCに分割した場合，データ構造に変更がなければ，変更が表示のみに限定され，該当するビュー(V)とコントローラー(C)のプログラムを変更すればよい。

また，今回の改修作業を通じて，さらにより効率化できる部分を発見した。それは，ビューのプログラムでボタンなどの位置とサイズや配色を設定する箇所で，ボタンのコンストラクタの引数の個数が多い部分である。引数の個数が多いのは，位置とサイズや配色が多数のパラメータから構成されているからである。アプリケーションを開発するのに使用しているProcessingはJavaがベースになっていて，Javaでは名前付き引数が使えないため，引数の個数が増えるとその順番を記憶する必要があるため，プログラマの負荷が大きくなる。引数にオブジェクトを使用してまとめ，パラメータの指定を分割することで，開発が容易になると思われる。

また，これは本案件の事情によるものだが，デザインに使用したMicrosoft PowerPointの長さの単位はcmで，プログラムではこれをピクセルに換算する必要があった。換算が必要な値の個数は多く，OSやPowerPointのバージョンによって換算方法が変わってくるので，プログラムの中でさまざまな場所に分散して記述すると管理が難しくなる。そこで，cmをピクセルに換算する関数を用意して一元管理すると開発が容易になると思われる。

参考文献

1) Reas, C., Fry, B., 中西 泰人, 安藤 幸央, 澤村 正樹, & 杉本 達應 (2015). Processing. ビー・エヌ・エヌ新社.
2) 小久保 温, 伊藤 智也, 小玉 成人 (2020). Android アプリ「はちのへ先人診断」の開発. 八戸工業大学紀要 第 39 巻 94-100.
3) 小久保 温 (2020). Android アプリ「はちのへ先人診断」の開発. 八戸市博物館研究紀要 第 33 号 66-51(ページ逆順).
4) Reenskaug, T. (1979). Mvc xerox parc 1978-79. Trygve/MVC.

要　旨

「南部昔ッコ診断」アプリは，「はちのへ先人診断」アプリのコンテンツを入れ替えて開発した。「はちのへ先人診断」の開発にはおよそ2ヶ月かかったが，「南部昔ッコ診断」の開発に要した期間はおよそ2日間であった。これはアプリをデータとプログラムに分割し，さらにプログラムを MVC に分割し，アプリに変更があった場合にもその影響範囲が狭くなるようにしたからである。コンテンツの入れ替え作業は，主に「南部昔ッコ診断」のグラフィックデザインの制作であった。コンテンツの入れ替えが，データだけを変更することでできることが理想的である。しかし，画面を司るビューのプログラムも変更する必要がある。今回のコンテンツの入れ替えから得られたビューの変更を容易にする知見を論じる。

キーワード：ソフトウェアの改修の容易性, MVC, Androidアプリ