広島経済大学研究論集
第37巻第4号　2015年3月

スマートデバイス用アプリ開発教育における アジャイルとデザインパターンの適用

伊　藤　則　之*

1. はじめに

ここ数年，スマートフォンやタブレットなどのスマートデバイスは世界的に広く普及し始めている。図1は2013年に発表された出荷台数の予測（資料[1]の28ページの図から傾向を示す図として筆者が作成）を示したものであり，2010年にAの部分ではスマートフォンがパソコンの出荷台数を超え，その後2015年にはBの部分ではタブレットがパソコンの出荷台数を超える予測となっている。さらに，パソコンの出荷台数が年々減少する中で，特にスマートフォンの出荷台数はCの部分に示すように急速に伸びる予測となっている。

スマートフォン市場において，2007年にAppleが米国を中心にiOSを搭載したiPhoneを発売し[2]，2008年にはHTCがGoogleのAndroid OSを搭載したHTC Dreamを開発して，T-Mobile USAがT-Mobile　G1として米国で出荷を始めた[3,4]。このようなiOSとAndroid　OSの登場とともに，スマートフォンの出荷が大きく伸び始めた。

生活の中でパソコンやスマートデバイスがどのくらいの時間使われているかというComscore社による2014年1月の米国での調査によると[5]，インターネット利用の約55％がスマートデバイスによるものであり，残り約45％がパソコンによるものとされている。スマートデバイスによる利用時間がパソコンによる利用時間を上回ったのは調査以来2014年が初めてとされている。日本においても，スマートフォンとパソコンのどちらをより多くの時間使うかという調査が日経BP社により20～40歳代のスマートフォン所有者500人に対してなされ[6]，この調査結果では35歳以上の男性の約60％がパソコンの利用のほうが長く，逆に35歳未満の女性の約60％はスマートフォンの利用のほうが長いという結果になっている。

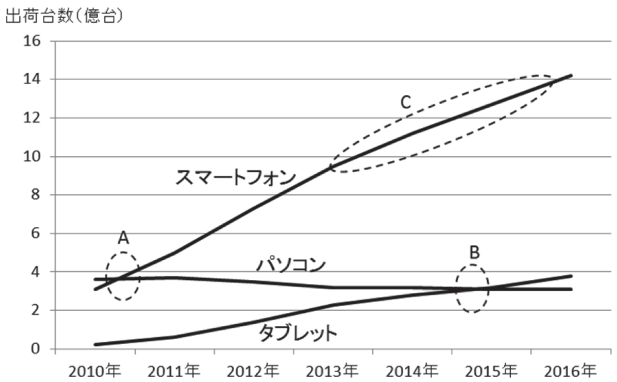


図1　スマートフォン，タブレットの出荷台数予測

* 広島経済大学経済学部教授

また，スマートデバイスが生活の中でどのように使われているかという調査が行われ，それによると2014年ではスマートデバイスを使う平均時間は１日当たり２時間42分で，そのうち２時間19分は Web 閲覧以外のアプリを利用していると報告されている[7]。つまり，スマートデバイスを使う１日当たりの平均時間の約80％をアプリの利用に費やしていることがわかる。

このように近年では，スマートデバイスにおけるアプリの利用が拡大しており，こうしたアプリの開発を教育現場において取り入れて行くことも重要となっている。本論文では，学校教育の中でのアプリ開発教育の重要性について考えた上で，教育現場でのどのようなアプリ開発教育がなされているかを調査し，本学のビジネス情報学科のゼミの中で実際に取り入れたアプリ開発教育に関する新たな取り組みを述べ，それについて考察を行う。

2.　アプリ開発教育の必要性

社会の中でスマートデバイスのアプリ利用が広がっている中で，大学などでの研究成果を実用化する手段としてアプリが利用されるケースもある。2011年に発生した東日本大震災の後，避難支援や防災教育のための研究成果をアプリで実現する例も多くなっている。たとえば，もし津波が発生した場合に津波がどれほど高くなるのかを実感してもらうために，実際の光景に津波の高さを表示するアプリ「津波 AR」が宮城教育大学と東北大学により開発されている[8]。このアプリは拡張現実（Augmented Reality: AR）の研究成果を防災教育用のアプリとして実用化したものである。

また，福祉支援の研究成果をタブレット用アプリとして実現している例[9]や，中学の理科教育において複数の滑車を組み合わせたときの動きを実際に滑車を組み合わせて実験しているかのように体験できるタブレット用アプリの例[10]，さらにタブレットでの手書きの操作性を向上させるためのユーザーインターフェース実現の例[11]などがある。

さらに近年では，国や自治体が持つ公共的なデータに対して，著作権や機密保持などの制約を付けずに，プログラムが読み込むことができる形式で広く公開するオープンデータという仕組みができ始めており，総務省では2012年よりこのオープンデータの実証実験を始めている[12]。この動きに合わせて，地方の自治体でも同様にオープンデータとして情報を公開して，新たな産業創出などを図っているケースもある[13]。福井県鯖江市の例では，バスの現在位置，公園のトイレ位置，自動体外式除細動器（AED）の配置場所など約40のオープンデータを公開して，これらを活用して民間が開発したアプリが80を超えたとのことである。

総務省による実証実験においては，オープンデータをスマートフォンなどのアプリでも有効活用して社会に役立てようということを目的に，2014年初めに「オープンデータ・アプリコンテスト」を実施している[14]。このコンテストの応募要項には，応募資格として「中学生以下の方が応募する場合は，保護者又は監督者（学校の先生等）の許可を得てください」という記述がある通り，中学生以下でもアプリを作成するケースを想定している。なお，このコンテストの受賞者には，企業や個人だけでなく，高校や大学からの応募者も含まれている。

このように，スマートデバイス用のアプリ開発は，特別な専門技術を持っている人だけが開発できるということではなく，中学生以下でもアプリ開発は可能な状況にある。アプリ開発を小中高生にも広げるために，ライフイズテックは，中学や高校向けに iPhone 用アプリの作成方法を学ぶことができる教材を無償で提供することを2014年に発表した[15]。これを受けて，中高一貫教育の品川女子学院が2015年度から導入

する予定とのことである。DeNA と東洋大学が共同で，佐賀県武雄市内の小学１年生を対象にタブレット用アプリ開発の教育の実証研究を2014年10月から始めるという事例もある[16)]。また，サイバーエージェントも2013年10月から小学生向けのアプリ開発塾の事業を始めている[17)]。

アプリ開発を若い年代から教育するために，「アプリ甲子園」と呼ばれるコンテストも開催されている[18)]。このアプリ甲子園は2011年から開催されており，現在では文部科学省が後援となって支援を行っている。この「アプリ甲子園」への参加者は最初の2011年には20名ほどであったが，2014年には約800名まで増えており，これは学校現場でも年々アプリ開発への関心が高まっていることを示している。なお，2014年は約800名が予選に参加し，その中の11名が決勝に進んだが，そのうちの１名は小学校５年生ということである。

このようにアプリ開発の教育を小学生や中学生の段階から始めようとする動きは，文部科学省が2012年の中学校の新学習指導要領において，「技術・家庭」の授業では従来選択科目であった「プログラムと計測・制御」という内容が必修となったことも影響していると思われる。また，政府が2013年６月に閣議決定した成長戦略「日本再興戦略」の中で，「義務教育段階からプログラミング教育を推進する」と明記されているということであり[19)]，アプリ開発教育の年齢的前倒しもこうした動きと呼応していると考えられる。

独立行政法人である情報処理推進機構が2013年に IT 企業約3,000社に実施した IT 人材調査の結果をまとめた「IT 人材白書　2014」[20)]において，今後３年程度の間に新規・拡大を予定している事業は図２（資料[20)]の42ページの図2-1-26から筆者が作成）に示すようにスマートフォン・タブレットアプリ開発が１番高い率となっている。また，現在実施している事業のうち人材不足を感じている事業も図３（資料[20)]の43ページの図2-1-28から筆者が作成）に示すようにスマートフォン・タブレットアプリ開発

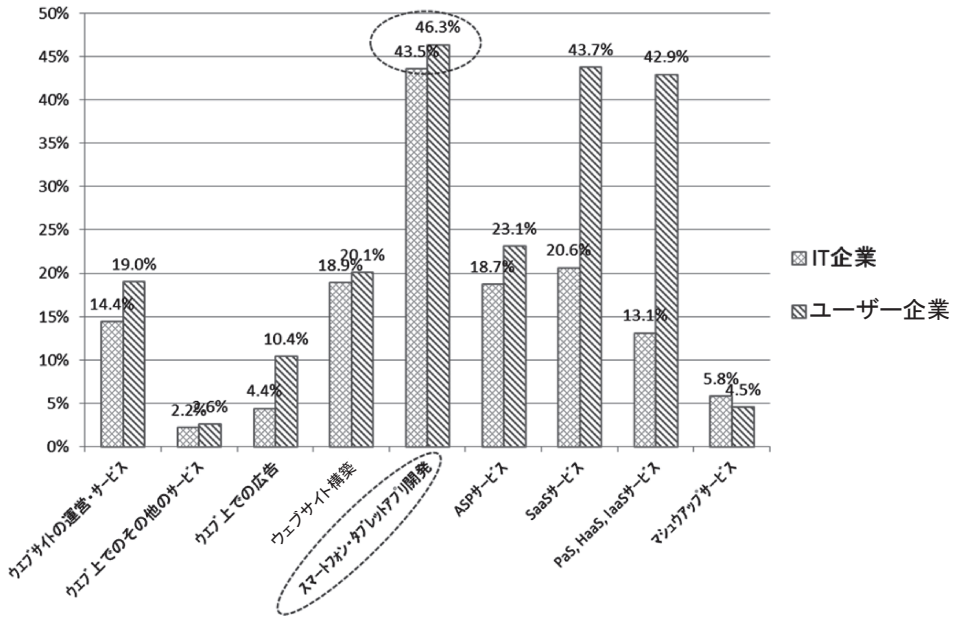


図２　今後３年程度の間に新規・拡大を予定している事業

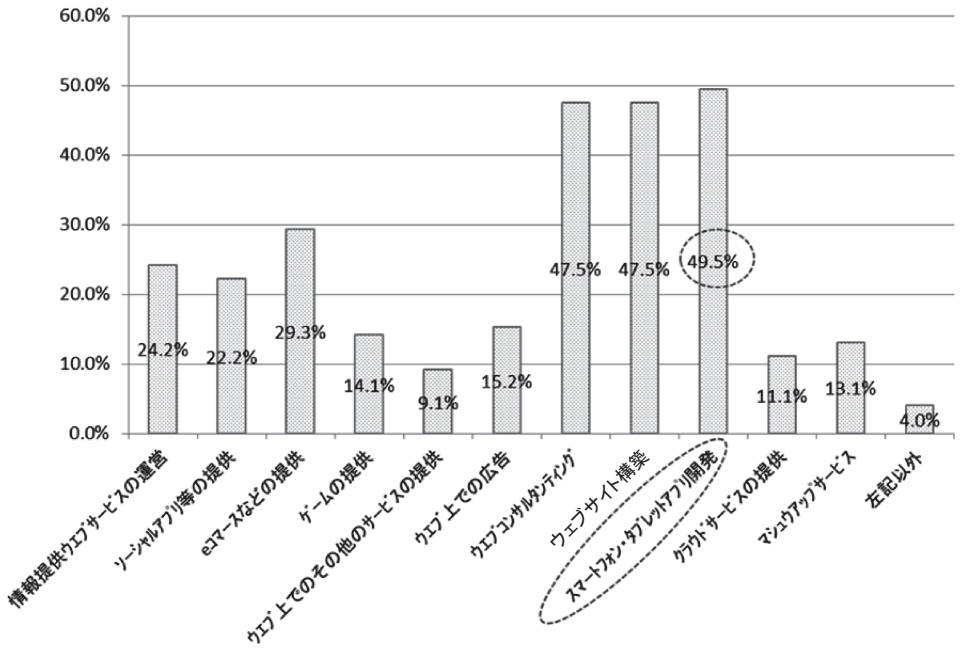


図３　現在実施している事業のうち人材不足を感じている事業

が１番高い率となっている。

このようにIT企業でのスマートデバイス用アプリ開発のための人材に対する不足感がある中で，地方自治体においてスマートデバイス用アプリ開発に関連するプロジェクトなどを立ち上げて，地元IT企業の支援や新規事業の展開を図ると同時に，人材育成を目指す例も見られるようになっている。岐阜県情報産業課は，2013年に「GIFU・スマートフォン・プロジェクト」を立ち上げ，スマートフォンを核として地域での情報化推進を進めている[21)]。青森県商工労働部新産業創造課は，「青森県における新産業創造への挑戦」というかたちで，2013年よりスマホアプリ技術人材育成研修事業を実施している[22)]。また，福岡県飯塚市では，2014年に独自に「e-ZUKA スマートフォンアプリコンテスト」を実施して，「IT技術者が集まる飯塚」を目指した取り組みを行っている[23)]。

3.　アプリ開発に関する学校教育の現状

一般的なプログラミングに関する教育についての研究はこれまで多数報告されている。しかし，スマートデバイス用アプリを開発するためのプログラミングは，アプリが動作するプラットフォームがスマートデバイスに限定されているという点において，一般的プログラミングよりもっと限定された条件のもとで教育が可能となる。こうしたスマートデバイス用アプリ開発に関する学校教育の研究はまだ多くないが，いくつか報告されている。

論文「スマートフォンのアプリ開発の授業について」[24)] では，秋田県立仁賀保高等学校の情報メディア科において，Android 搭載のスマートデバイス用アプリを簡単にプログラミングするツールとして現在ではマサチューセッツ工科大学のメディアラボから無償で提供されている App Inventor を使って教育を行っている例が報告されている。この App Inventor は，キーボードで文字をタイピングするようなプログラミンを必要とせず，マウスでブロックを組み合わせてプログラミングを行うことができることが特徴で，初心者でもアプリを簡単に作ることがで

きるツールである。しかし，画面の大きさや機能が限定されているために，本格的なアプリを作ることには向いていない。しかし，プログラミングが分かりやすいために，アプリ教育の導入として使われている例も多い[25, 26]。本学でも教育・学習支援センターの企画により昼休み時間帯を使って2013年度に実施した「ひるがく講座」において，筆者も週１回20分で１つのアプリを作る計５回の講座においてこのツールを利用した。図４は，この講座で学生に作ってもらったお絵かきアプリの例であり，図の（a）のように画面のデザインを行い，（b）のようにブロックを組み合わせてプログラミングを行い，そして（c）のように仮想的なスマートデバイスにアプリを転送して動作を確認することができる。ツールの立ち上げなどの準備を事前に行っていれば，約20分間で完成可能なアプリである。

論文「Android スマートフォンを用いたアプリケーション演習教材の開発」[27] では，北海道工業大学創生工学部・情報フロンティア工学科において，１年生を対象にスマートフォン用アプリ開発を教育するために，プログラミング言語の知識がなくてもアプリ開発が可能となる教材について提案を行っている。この教材では，Android 用アプリ開発の際に一般的に利用される Eclipse という統合環境を使いながらも，アプリの画面に学科の説明を表示するための画面デザインを部品の配置と属性（幅，高さ，背景色など）を変更するだけで完成できるように工夫している。また，学生が興味を持つように，カメラや GPS の機能を使い，あらかじめ登録したマーカーをカメラを通して認識して，マーカーに対応して情報を画面上に表示する拡張現実タイプのアプリ開発教材を作っている。この例では，十分準備された教材を使うことにより，プログラミング言語を知らなくてもアプリを動かして学生が達成感を得られるように工夫している。

論文「SPI テスト　トレーニングのアンドロイドアプリの試作」[28] では，大学生が就職活動の中で受ける必要が出てくる SPI テストとい

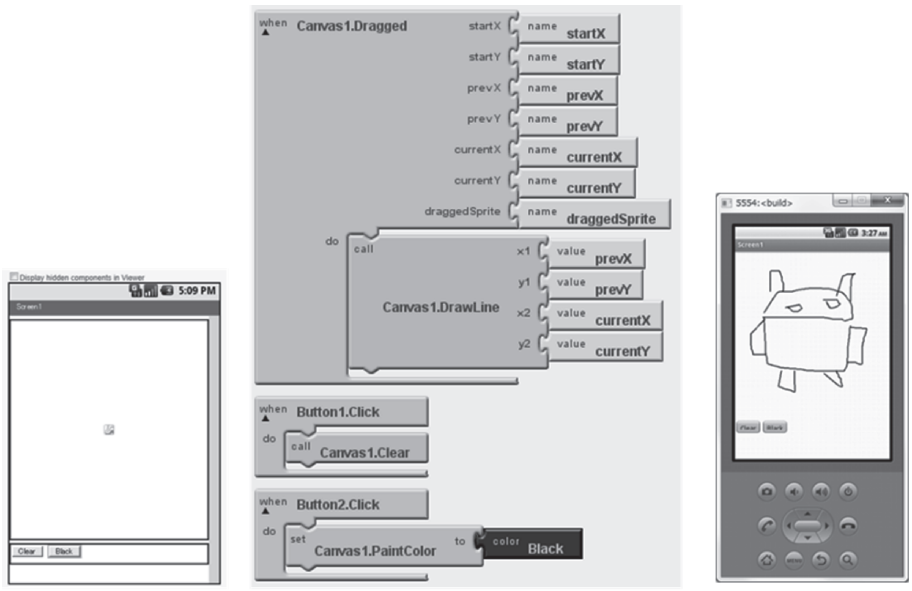


図４　App Inventor を使ったお絵かきアプリの作成

う社会人基礎力テストの対策を題材として，そのためのトレーニングアプリを作るための教育環境と手法に関する報告である。この例では，スマートデバイス用アプリをハイブリッドアプリとして開発している。スマートデバイス用アプリを作る際，Android 用アプリを作る場合には Android 機能を組み込んだ Eclipse という開発環境で Java 言語を使うことが必要である，iOS 用アプリを作る場合には Xcode という開発環境で Objective-C 言語を使う必要がある。このようにアプリを動かすデバイスによってアプリの開発環境が異なっている。これに対してハイブリッドアプリとは，同じ開発環境で作った１つのアプリが iOS でも Android でも動作するアプリのことを意味する。Web ブラウザーで閲覧可能なホームページなどを作成する場合，HTML や CSS，また JavaScript など OS に依存しない言語で記述することが可能である。ハイブリッドアプリも考え方は，HTML や CSS，また JavaScript を使って，OS に依存することなく Web ブラウザーで動作するアプリを作ろうというものである。このように OS に依存せずに開発ができるというメリットがあるが，スマートフォンの中で利用できる機能が限定されたり，画像の高速な描画には向かないなどのデメリットもある。

論文「Using Mobile Phone Programming to Teach Java and Advanced Programming to Compter Scientists」[29] は，Middle Tennessee State University において，Android 機能を組み込んだ Eclipse という開発環境で Java 言語を使ったアプリ開発をコンピュータ・サイエンス専攻の学生に教育するための新たな授業計画を提案している。この授業を履修する学生はすでに C 言語を習得していることを前提に，合計14回の授業を５つのユニットに分け，最初の２つのユニットの計３回で Java 言語によるプログラミングを教育し，次の２つのユニットの計３回で Android 用アプリ開発を例題を使って教育する。そして，最後のユニットの計７回でグループ単位でアプリの企画から開発，そして最終的なプレゼンテーションまでを行うという内容になっている。

以上のように，スマートデバイス用アプリ開発の教育については，どのような開発環境でどのような題材を使うか，またどのような授業計画が効果的なという面での研究が多い。筆者のゼミの学生にアプリの開発手法を教育する中では，何を題材とするかや授業計画の内容ではなく，学生一人ひとりが自分で実際にアプリ開発ができるようにするためにはどのようなプログラミング教育が効果的であるかという観点での取り組みを行った。具体的には，Android 搭載のスマートデバイス向けアプリ開発の一般的な開発環境を使い，アジャイルおよびデザインパターンというプログラム開発における２つの手法をアプリ開発教育に適用している。次の章以降では，本学での筆者のゼミでの取り組みについて述べる。

4.　アプリ開発教育のための新たな手法の導入

ビジネス情報学科のゼミ制度は，２年生からグループ演習というゼミが始まり，同じ教員の下で３年生の卒業研究Ⅰ，４年生の卒業研究Ⅱと続く（本学での大学改革の結果，2015年度入学の学生からこのゼミ制度は全学のゼミ制度に統一されるため，２年生のグループ演習の部分が変更される）。そのため，学生は，１年生の後期に学科の各先生の研究室を訪問して，それぞれのゼミでの研究内容を聞きながら自分の学びたいゼミを選択する。筆者は，2012年より２年生のグループ演習において，アプリ開発をゼミ教育の中に取り入れている。

筆者のゼミでは，２年生から３年間の合計90コマの中で，同じ学生に継続的にアプリ教育を

表 1　筆者のゼミにおけるアプリ開発教育

学年とゼミ	年間のコマ数（1コマ90分）	教育内容
2年生 グループ演習	30	ゼミ学生共通のアプリ開発教育
3年生 卒業研究Ⅰ	30	各自が企画したアプリを開発
4年生 卒業研究Ⅱ	30	アプリを完成させて卒論作成

行うことができるというメリットを最大限に活かし，表1に示すように，2年生でアプリ開発の基本を教育し，3年生から始まる卒業研究の中で学生各自が企画したアプリの開発を行い，それぞれが自分のアプリについて卒業論文としてまとめるというかたちで進める。ビジネス情報学科ではアプリ開発のような情報技術だけでなく，ビジネス面での教育も重要となる。アプリを自分で企画して開発し，それをマーケットに出して，どのようにしてどのくらい収入が入るかという一連の実験も可能であり，ものを作る喜びとビジネスの楽しさの両面をゼミの学生に伝えることが私の最終的な目標である。このようにビジネスの要素も取り入れることにより，単にアプリ開発技術を高めるためだけでなく，そのアプリがどのように社会の人々に役に立つかを考えながら進めることができるため，教育効果が高いと考えられる。特に，Android 用スマートデバイス向けのアプリについては，Google が Google Play というアプリのマーケットを公開しており，初回1度だけ25ドルの手数料を払うことにより，誰でもアプリを個数に制約なく登録することができる。このように，アプリを流通させるマーケットが確立しており，個人でアプリを作成した場合でも簡単にこのマーケットにアプリを流通させることが可能である。アプリを有料にすることも，無料にして広告を付けることも可能であり，収益を上げるための環境も準備されている。このように，学生が自ら企画したアプリを開発し，それをマーケットに出して，その後も継続してメンテナンスしたり，場合によっては収益も考えることも可能であるため，筆者のゼミではアプリ開発はグループ単位とはせずに，各学生が独自のアプリを企画して開発するかたちで進めている。

ゼミでアプリ開発を行うための開発環境として，Android 機能を組み込んだ Eclipse という Android 用アプリを開発する際の一般的な開発環境を本学の情報センターに依頼して筆者のゼミ室の各パソコンに構築してもらった。2012年に Android 用アプリの開発環境を構築した際には，Window 7 の上に JDK 6＋Eclipse 3.6（Helios）＋Android SDK 17版＋ADT 17版という環境であったが，その後2013年度末に全学的にパソコンが Windows 8 に置き換わったために，JDK 7＋Eclipse 4.2（Juno）＋Android SDK 22版＋ADT 17版という環境に置き換えた。なお，Eclipse でのメニュー表示などを日本化するために Pleiades をプラグインしている。

2年生のゼミの学生が Android 用アプリ開発を習得するには，まず Eclipse という開発環境に慣れる必要がある。そのためには，図5のようにアプリ画面デザインを示しながら簡単な例題アプリの説明をゼミの学生に行う。なお，この例題では，画面のねこの写真をタッチするとねこの鳴き声が聞こえるというアプリであり，Android アプリ開発の入門で使われることが多い。この後，図6のようなソースコードが書かれた資料を配布し，その資料では Eclipse で学生が自分で入力する部分は下線付きで示し，Eclipse への操作により Eclipse が自動で生成してくれる部分は下線なしにしている。そして，

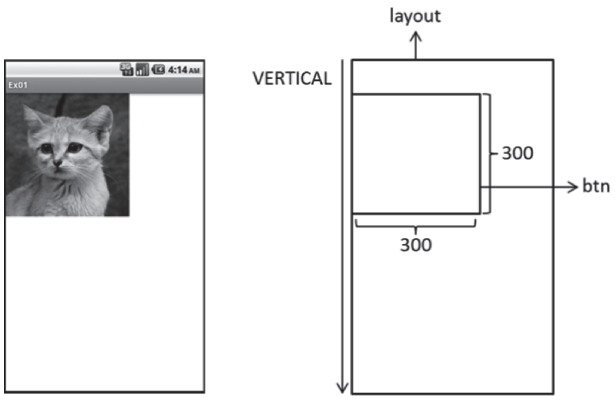


図５　Eclipse 操作練習のための例題アプリ

```
public class MainActivity extends Activity implements OnClickListener {

    private Button   btn;      // ボタンに自分でbtnという名前をつける
    private MediaPlayer   mp;      // MediaPlayerに自分でmpという名前をつける

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        LinearLayout layout = new LinearLayout(this);      // 部品の入れ物をlayoutとい名前で作成
        layout.setBackgroundColor(Color.WHITE);      // 背景を白にする
        setContentView(layout);      // 入れ物layoutを画面に表示する

        btn = new Button(this);      // btnという名前のボタンを作成する
        btn.setWidth(300);      // ボタンbtnの幅を300にする
        btn.setHeight(300);      // ボタンbtnの高さを300にする
        int id = getResources().getIdentifier("neko","drawable",getPackageName());      // neko.jpgの取り出し
        btn.setBackgroundResource(id);      // ボタンbtnの背景にneko.jpgの絵をセット
        btn.setOnClickListener(this);      // ボタンbtnがタッチされたら何か処理をすることを指定する

        layout.addView(btn);      // ボタンbtnを入れものlayoutに入れる
    }

    @Override
    public void onClick(View arg0) {      // ボタンがクリックされたとき、ここの処理が実行される
        // TODO 自動生成されたメソッド・スタブ
        mp = MediaPlayer.create(this, R.raw.cat);      // cat.mp3をrawフォルダから取り出して、mpにセット
        mp.start();      // mpをスタート
    }
}
```

図６　例題アプリのソースコード

下線付きの部分だけを入力してアプリのソースコードを作り上げる練習を何度か繰り返して操作方法を習得してもらう。また，図６のソースコードでは，各行の後に“//”で始まる部分に説明を入れており，学生が入力しながら各行が何を意味しているかを確認できるようにしている。

Eclipse の操作では，新規プロジェクトの作成，ソースコードの入力，画像や音声のファイルのコピー，仮想デバイスへのアプリの転送，そしてデータの保存までの一連のことを自由にできるようになるまで複数回繰り返してもらう。この操作に慣れるまでは，入力しているソースコードがどういう意味かについて細かな説明は行わない。Eclipse の操作に関する教育の中で特に重要と思われる項目は，次の３点であると考える。これらの３点が徹底されていない場合，問題が発生したときに学生自身が解決できないケースが多く，その解決のために教員による問題解決の時間が必要となり，新たな教育手法の

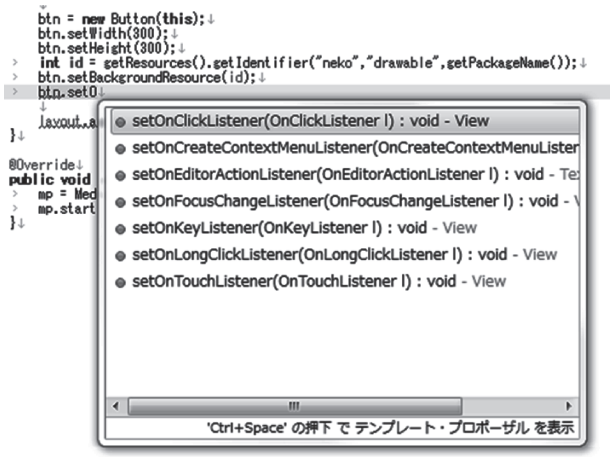


図 7　Eclipse におけるコードアシスト機能の活用

実践に支障が出てくるためである。

1）コードアシスト機能の積極的利用

学生によるキーボードでのタイプ量をなるべく少なくするとともに，タイプミスを少なくするために，プログラムの各行の先頭の数文字をタイプ入力した段階で“Ctrl”キーを押しながら空白キーを押すことにより，図 7 のような候補の一覧が表示される。この機能はコードアシストと呼ばれており，候補として表示されている一覧から適切なものを選択することが重要である。

2）中括弧“{”および“}”の位置的対応付け

Java 言語だけでなく多くの言語でプログラムを記述する際に重要なことの 1 つに，中括弧“{”および“}”の対応をわかりやすく記述することがある。Eclipse では，ソースコードで中括弧の初めである“{”を入力した直後に“Enter”キーをタイプインすることにより，初めの“{”に対応する終わりの“}”が自動で入力される。図 8 のように①のように“if”と“}”が垂直方向の同じ位置に来るようにするために，②を入力した直後に“Enter”キーを押すことにより，③が自動的に入力される。同様に，④のような位置になるように，⑤を入力した直後に“Enter”キーを押して⑥を自動で入力し，⑦についても⑧に対応する⑨を自動で入力するようにする。この操作の練習を徹底することにより，“{”と“}”は正しく対応しているため，何か問題が発生したときにソースコードを見ながら問題個所を容易に特定することができる。

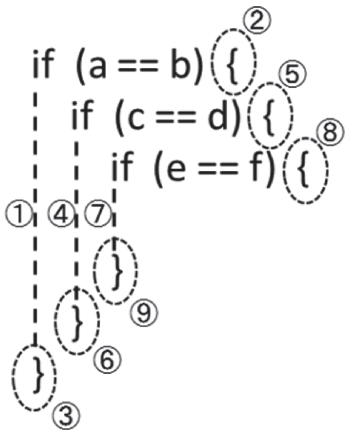


図 8　中括弧“{”および“}”の位置的対応付け

3）**LogCat** によるプログラム不具合箇所の特定

アプリ開発教育において，新しい手法を導入したとしても，例題アプリのソースコードが印刷された資料を学生に配布して，その資料を見ながらキーボードを使って入力してもらうことは必ず必要となる。しかし，前述の 1）や 2）を使ってソースコードの入力ミスを少なくしたとしても，資料に印刷してあるソースコードの

ある行を入力し忘れるということもしばしば発生する。その場合，Eclipse 環境では構文ミスがなければ何もエラー表示されず，アプリの実行へと進むことができる。その場合，アプリ実行と同時に図９のようにエラーが発生して，アプリが正しく起動されないことになる。このようなエラーを毎回教員が調査することは時間的に非常に無駄であるため，学生自身がなるべくこのエラーを解決できるようにしておくことが重要となる。そのために，実行時にエラーが発生したとき，ソースコードの何行目でエラーが発生しているかを学生自身が見つける方法を学んでもらうことが必要である。

アプリの起動後にエラーが発生した場合，LogCat という画面にエラーの場所と原因が表示される。図10は LogCat 画面の一部を示している。画面下から順に上の方へとエラーのメッセージを確認し，各行の“(”と“)”の中に自分のプログラム名と同じ個所を探す。この例では，MainActivity.java というソースコードを実行しているので，図10を下から見ながら各行の“(”と“)”の中に MainActivity.java と表示されている行を探す。図10の丸印で囲まれた部分が該当する箇所であり，MainActivity.java の後ろのコロン記号の次にエラーが発生している行番号が示されている。この例では26行目がエラー個所となるので，Eclipse のソースコードの画面でなぜ26行目がエラーになったかを学生に説明しながら，エラー個所の見つけ方を教育して行く。

図９　アプリ起動時のエラー

こうして，Eclipse 操作に学生が十分慣れたら，いよいよアジャイルおよびデザインパターンというプログラム開発における２つの手法をアプリ開発教育に適用する。

4.1　アジャイル開発をベースにしたアジャイル的教育法

ソフトウエアの開発手法はいくつか存在するが，その中で代表的な２つはウォータフォール開発とアジャイル開発である。ウォーターフォール開発では，これから作るソフトウエアに必要な機能のすべてを要件定義というフェーズで決定し，決定された要件すべてを実現するようにプログラム構造の設計を行う。この設計に基づいてプログラミングを行い，最終的にすべての機能が動作するかをテストして，ソフトウエアの完成となる。ウォーターフォール開発では，滝の流れのように各フェーズを順番に実行しながら，テストという最終フェーズが終了するとようやくソフトウエアが完成するので，作り始めてから動作させるまでの期間が長くな

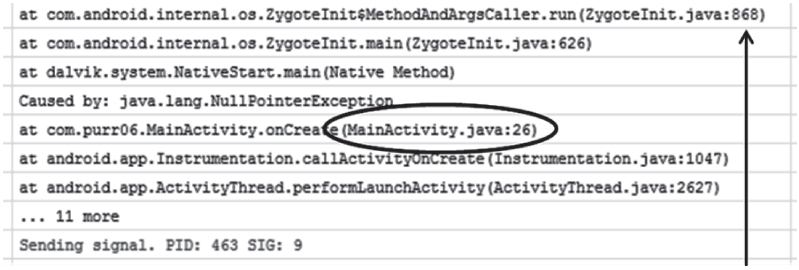


図10　Eclipse の LogCat でのエラー個所の特定

る傾向にある。これに対してアジャイル開発では，すべての機能の中で１番最初に必要となる１つの機能について，要件定義を行い，プログラム構造を設計し，プログラミングを行い，機能をテストする。最初の機能が完了したら，次に必要となる機能について同様な繰り返しを実行する。このような繰り返しを続けながら，必要な機能から順次追加して行く。

この２つの手法は，重視するものが極端に異なっている。ウォーターフォール開発はソフトウエアの古典的か開発手法であり，ソフトウエアの利用者と約束した契約や計画を重視する開発手法である。そのために，ソフトウエアを作り始める前に，作るべき機能をすべてリストアップし，それぞれの機能の仕様を決め，さらにはスケジュールや費用なども決めた上で，作る側と利用する側がしっかり合意してから作り始める。この手法では，すべての機能をまとめて作るため，作るための契約を合意してから完成するまでの期間が長くなる。完成したソフトウエアを使うことができるのは，作り始めてから数か月や数年経ってからということも珍しくない。このようにしてでき上がったソフトウエアは，利用者が想定していたものとかなり異なるものになっているということがしばしば発生する。このようにして，ウォーターフォール開発では契約や計画を守って作ったにもかかわらず，利用者の立場から見ると失敗作ということが発生してしまう場合もある。これは，ソフトウエア開発の最初のフェーズですべての機能について仕様を確定すること自体に無理がある場合が多いからと思われる。実際，ソフトウエア開発に関係する380人にアンケートしたある調査では，57％の人が失敗を経験したことがあると回答しており，その原因の第１位は「仕様が固まらない，あいまい」とのことである[30)]。

一方，アジャイル開発は2000年代に入ってから提唱された新しい開発手法であり，とにかく利用者に取って満足できる動くソフトウエアを作ることを重視する開発手法である。つまり，アジャイル開発は，契約や計画ではなく，適用や動作を重視する開発手法である。この開発手法は，2001年の２月に17名のソフトウエア研究者がアメリカのユタ州にあるスキーリゾート地に集まり，「アジャイルソフトウエア開発宣言」としてまとめたものであり，現在でも Web 上に多くの言語に翻訳されて公開されている[31)]。この宣言の中では，ソフトウエア開発において，プロセスやツールよりも個人と対話を，包括的なドキュメントよりも動くソフトウェアを，契約交渉よりも顧客との協調を，計画に従うことよりも変化への対応を価値とするとしている。こうした価値を実現するために，まず最初に必要とされる機能に限定してできる限り短期間で開発し，それを利用者に使ってもらい，利用者からの要望があればすぐに取り入れるというかたちで開発を続ける。このようにすることで，随時利用者の要望を取り入れることができ，しかも機能単位ですぐにテストを行うことができるため，意図した通りに動作しない場合でも，その原因を探すプログラムの範囲は小さいので，容易に原因を見つけることも可能となる。

このようなアジャイル開発の特長は，アプリ開発の教育にも適用が可能であると考える。ある例題とするアプリについて，学生にそのソースコードを示し，アプリの機能をソースコード上ではどのように実現しているかを説明する場合を想定する。多くの機能を含むアプリに対応するソースコード全体を見せて，この機能はソースコードのこの部分で実現しているというように説明したのでは，その機能に関係しないソースコードの記述も一緒に目に入ってしまうので，学生に取って理解することが難しくなる。これは，ウォーターフォール開発を適用してでき上がったすべての機能を含むソフトウエアにおいて，ある特定の機能の動作を確認する際，

もし動作しなかったときに原因を究明する際はその機能に関係しない記述を含むソースコード全体を調査しなければならず，調査が難しいことと本質的には同じである。

そこで，アジャイル開発に模したかたちでアプリ開発教育も行う。図６に示されたソースコードは，画面に表示されたねこの画像をタッチするとニャーという音声が再生されるアプリであり，これを例に取ってその教育方法を具体的に示す。アプリを作り慣れた人に取っては，図６のソースコードはとても簡単に理解できるレベルのものである。しかし，初心者に取っては英語の羅列に見えて，非常に理解しにくいものである。そこで，このアプリをどのように実現しているかをソースコード上で説明する際には，そのアプリの機能を表２に示すように分解して，各機能を段階的に説明し，それぞれの段階で新たに追加する機能に対応する記述をソースコードに自分で入れて，アプリの動作確認まで行ってもらう。つまり，これが，アジャイル開発の手法をそのまま教育に導入したアジャイル的教育である。

表２　例題アプリの機能分割と段階的説明

段　階	機　　能
ステップ１	画面に１つのボタンを表示
ステップ２	画面の背景の色を白を変更する
ステップ３	ボタンのサイズを大きくする（300×300）
ステップ４	ボタンにねこの写真を入れる
ステップ５	ボタンをタッチしたら「ニャー」という文字が表示される
ステップ６	ボタンをタッチしたら「ニャー」という音声が出る

図11は表２のステップ１に対応するソースコードであり，記述量も少なく，アプリ開発を初めて経験する学生に取っても，十分理解できるレベルである。次に表２のステップ２に対応するソースコードを学生に提示する際には，ステップ２で追加された部分が明確にわかるように図12のように下線を付加するか色を変えて提示する。図13は，表２のステップ３に対応して学生に示すソースコードであり，それぞれのステップで追加されるのはこの例では１～２行である。このように段階的に機能を追加して，それぞれの段階で機能の動作確認を行うことによって，ソースコードの記述とアプリの動作の対応が容易にわかるため，学生に取っては非常に理解しやすくなる。また，もし機能が意図した通りに動作しないと申し出た学生がいる場合，学生本人に追加した部分を再度確認するように促すことにより原因を特定しやすくなり，もし学生が自分で原因を特定できなかった場合でも，指導する教員も見るべきソースコードの部分が限定されるので，原因特定が容易になる。

```
public class MainActivity extends Activity implements OnClickListener {

    private Button   btn;     // ボタンに自分でbtnという名前をつける

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        LinearLayout layout = new LinearLayout(this);     // 部品の入れ物をlayoutとい名前で作成
        setContentView(layout);     // 入れ物layoutを画面に表示する

        btn = new Button(this);     // btnという名前のボタンを作成する

        layout.addView(btn);     // ボタンbtnを入れものlayoutに入れる
    }
}
```

図11　例題アプリのアジャイル的教育の例（ステップ１）

```
public class MainActivity extends Activity implements OnClickListener {

    private Button   btn;     // ボタンに自分でbtnという名前をつける

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        LinearLayout layout = new LinearLayout(this);     // 部品の入れ物をlayoutとい名前で作成
        layout.setBackgroundColor(Color.WHITE);     // 背景を白にする
        setContentView(layout);     // 入れ物layoutを画面に表示する

        btn = new Button(this);     // btnという名前のボタンを作成する

        layout.addView(btn);     // ボタンbtnを入れものlayoutに入れる
    }
}
```

図12　例題アプリのアジャイル的教育の例（ステップ２）

```
public class MainActivity extends Activity implements OnClickListener {

    private Button   btn;     // ボタンに自分でbtnという名前をつける

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        LinearLayout layout = new LinearLayout(this);     // 部品の入れ物をlayoutとい名前で作成
        layout.setBackgroundColor(Color.WHITE);     // 背景を白にする
        setContentView(layout);     // 入れ物layoutを画面に表示する

        btn = new Button(this);     // btnという名前のボタンを作成する
        btn.setWidth(300);     // ボタンbtnの幅を300にする
        btn.setHeight(300);     // ボタンbtnの高さを300にする

        layout.addView(btn);     // ボタンbtnを入れものlayoutに入れる
    }
}
```

図13　例題アプリのアジャイル的教育の例（ステップ３）

このようなアジャイル的教育においては，このアプリを自分自身ではない仮想的な利用者を想定して開発することを学生に説明し，ステップ１での機能をまず開発して利用者に示したら，画面の背景は黒ではなく白がよいという要望があって追加を行い，そのあとボタンのサイズ，そして背景の画像，最後には画像をタッチしたら音声が鳴るというように要望が順次出てくるというかたちで，仮想的な利用者と対話するようなイメージを持たせることにより，アジャイル開発手法そのものについても同時に学ぶことができる。例題とするアプリをどのくらい細かい機能に分割してアジャイル的教育を適用するかについては，最初はできるだけ細かく分割し，次第に理解が進んだ段階では機能の分割を少し粗くするなど，学生の理解度を見ながら調整して行くことが必要である。２年生のグループ演習というゼミの前半では，このようなアジャイル的教育をベースにして，アプリ開発において機能を実現するために記述するソースコードをまず理解してもらう。こうした基本的なことが理解できている２年生の後半では，さらに4.2で述べるデザインパターンをベースにしたアプリ開発の教育により，自分で企画したアプリを開発できる力を身に付ける教育を行う。

ウォータフォール開発とアジャイル開発では，ソフトウエア開発における考え方が異なっており，筆者のゼミにおいてアプリ開発を教育する際には，アジャイル開発の考え方がより適していると考えている。ウォータフォール開発とア

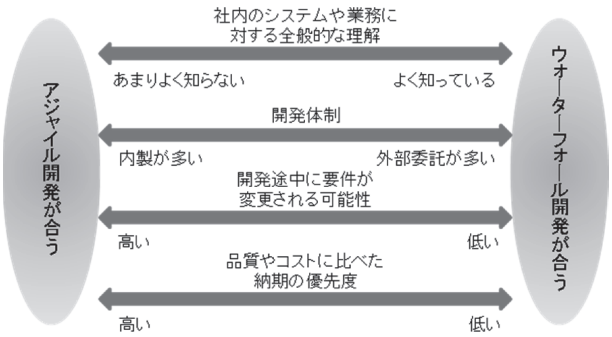


図14　ウォータフォール開発とアジャイル開発の使い分け

ジャイル開発という２つの開発手法は，どちらの開発手法がよくて，どちらの開発手法が悪いと決めるべきものではなく，開発するソフトウエアではどちらの手法がより適しているかを考えて判断し，選択すべきものである。たとえば，文献「ウォーターフォール開発と使い分け」[32]では，図14に示すような使い分けが示されている。３年生の卒業研究からゼミの学生は各自でアプリを企画してそれを自分で開発することになる。３年生でのアプリの作り始めの段階では，これから自分が作るアプリについてまだ仕様が明確になっておらず，開発するのは自分自身であり，開発途中で機能などの要件が変更される可能性は非常に高く，品質よりもまず期限までに動作させることの優先度が高い。そのため，図14の使い分けに照らし合わせても，３年生から始まるアプリ開発ではアジャイル開発が適している。したがって，２年生のグループ演習の段階からアジャイル的教育を実践することにより，３年生でのアプリ開発でもアジャイル開発の手法が学生に自然と受け入れられる。実際に３年生のゼミにおいて，１度にまとめて複数の機能をソースコードに記述した学生が，アプリを動作させたら動かなくなり，どこが原因かがわからないという相談があった。その際には，再度１つの機能を入れた段階でテストしながら進めるように指導して，学生もその方法に納得した例があった。

4.2　デザインパターンをベースにしたアプリ開発の教育

デザインパターンとは，オブジェクト指向のプログラミングにおいて用いられるプログラミング技法の１つである。この考え方は，1987年のオブジェクト指向プログラミングに関するコンファレンスにおいて発表された「Using Pattern Languages for Object-Oriented Programs」という論文[33]が起源と言われている。この論文では，建物の建築家であるAlexanderという人のパターン言語という考え方をプログラミングにも適用しようと提案している。Alexanderは，プロの建築家は，これまで設計した建物に取り入れた設計をパターンとして数多く持っており，新たな建物を建築する際にはこうして持っているパターンを組み合わせることが有効だと指摘している。

実際にプログラミングにおいても，すでにでき上がっているパターンを教育して，それらを組み合わせることにより，そのパターンがどのようにしてできているかということを理解しなくても，目的のプログラムが短い日数で完成できることをその論文では示している。つまり，デザインパターンとは，これまでのプログラミングのノウハウを再利用できるかたちにして，それらをまとめてカタログ化したものと言うことができる。現在では，1995年に出版された「Design Patterns: Elements of Reusable Object

Oriented Software」という書籍[34]で示されている23個のデザインパターンが有名となっている。なお，本論文では，これまでのプログラミングのノウハウを再利用できるかたちにしたものという一般的な意味で使っている。

筆者のグループ演習の中でも，２年生のグループ演習で学生が実際に作成したすべての例題がデザインパターンとして有効に使えるようになることを目指している。表３は，2014年度の２年生のゼミで利用したデザインパターンの例を示す。この例では，デザインパターン１から11までの段階では，またそれらのパターンを組み合わせて新たなアプリを作るというような応用する方法の教育は行っていない。この理由は，デザインパターン１から11までは１つの画面だけを持つアプリであり，機能も非常に限定されているために応用問題を作りにくいためである。デザインパターン12では，初めて複数の画面を持ち，それらの画面間を自由に遷移できる仕組みを教育する。このように複数の画面を持つことでアプリの機能としての自由度が増すために，この段階でこれまでに学んだデザインパターンを利用して，それらを組み合わせるアプリを応用問題として学生に提示して，自分で考えながらアプリを作り上げる方法を教育する。

このようなデザインパターンをベースとしたアプリ開発の教育をできるだけ効果的なものにするために，２つの工夫を行った。１つ目の工夫は，アプリの画面デザインをすべてプログラムのソースコードの中で行うということである。Android 用スマートデバイス向けアプリ開発のプログラミング環境では，画面デザインに関して，Java 言語で記述されたプログラムとは別に XML 言語で記述する方法とプログラム内部で一緒に Java 言語で記述する方法がある。アプリ開発の一般的な考え方としては，画面デザインは XML 言語で記述して，Java 言語によるプログラム記述とは分離したほうがよいという考え方もある。XML 言語で記述できることが Java 言語のよるプログラム記述ではその記述

表３　2014年度の２年生のゼミで使ったデザインパターンの例

デザインパターン番号	内　　容
1	画像付のボタンをタッチすると音声ファイルを再生
2	ボタンをタッチすると入力したテキストを読み上げ
3	URL を入れてボタンをタッチすると WEB ブラウザーが起動
4	アプリの中に持つ動画を再生
5	ボタンを複数並べて，タッチされたボタンの番号を表示
6	ボタンがタッチされると表示窓にランダムな数字を表示
7	３つの表示窓にランダムな数字が変化して停止
8	ボタンをタッチするとアニメーションを表示
9	指定した日付や時間を取り出して画面に表示
10	３，２，１，０と画面上にカウントダウンを表示
11	現在から指定された日付までの日数を計算して表示
12	ボタンのタッチによる複数の画面間で遷移
13	複数のボタンを表示し，タッチされたら対応するアプリを起動
14	SD カードへのテキストファイルの書き込みと読み出し
15	画面上への複数イメージボタンの配置とタッチ動作への反応

方法がない例もいくつか存在しているし，多くの解説書では XML 言語で画面デザインを行っている。また，直接 XML 言語を記述せずにグラフィカルに画面をデザインするユーザーインターフェースを利用することもできる。しかし，アプリを開発している人々の中でも，実際どちらのほうがよいのかが議論されているが，決定的にどちらがいいという結論にはなっていない[35, 36]。

アプリ開発の経験がない学生にデザインパターンをベースにアプリ開発手法を教育する場合，XML 言語で記述されたデザインパターンと Java 言語で記述されたデザインパターンの２種類を理解してもらうことより，Java 言語で記述されたデザインパターンのみを理解してもらうほうが効果的であると筆者は考える。図15は，図６の例題アプリにおいて，画面デザインを XML 言語で行い，その他の機能を Java 言語でプログラム記述した場合の例を示す。この例では，Java 言語のよるプログラム記述部分が多少少なくなるものの，XML 言語と Java 言語の合計行数で比較すると Java 言語だけで記述した図６の場合より多くなっている。さらに，画面デザインを XML 言語で記述した場合，XML 言語で記述されたボタンを Java 言語で記述されたプログラム内部では findViewById（R.id.button1）のような記述で“button1”という XML 言語記述の中でボタンに付けられた名前をもとに対応付けを行う必要があり，Java 言語で記述されたプログラムの読みやすさが損なわれてしまう。そのため，表３に示すデザインパターンを教える例題では，XML 言語を使

[XML 言語による記述]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
    <Button android:id="@+id/button1"
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:background="@drawable/cat"></Button>
</LinearLayout>
```

[Java 言語による記述]

```
public class MainActivity extends Activity implements OnClickListener {

    private Button bt;
    private MediaPlayer mp;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bt = (Button) findViewById(R.id.button1);
        bt.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        mp = MediaPlayer.create(this, R.raw.cat);
        mp.start();
    }
}
```

図15　図６の例題アプリを XML 言語と Java 言語で記述した場合の例

わずに，画面デザインも Java 言語で記述することを選択している。

2つ目の工夫は，プログラムのソースコードを説明する際に，デザインパターンを構成するパーツ（部品）毎に機能の説明を付与し，パーツによるデザインパターンの構成がわかるようにすることである。図16と図17は，表3におけるデザインパターン14におけるテキストファイルの書き込みの部分の抜粋であり，説明の入れ方を変えた2つの例である。図16はデザインパターンとしてあとで利用されることを想定せずに，またデザインパターンを構成するパーツを

```
// view オブジェクトの種類毎の処理を行う
// ボタンはプログラムで作り出しているので、R.id.button1 とかは使えないので、下記のようになる
if (v == bt[0] ) {      // Read ボタンがクリックされた時の処理
    try {      // ファイルへの入出力など例外が発生する可能性のある場合は try で囲む
        // 入力ストリームの生成。ファイル名は変数 name に入っている
        FileInputStream inputStream = openFileInput(name);
        // 入力ストリームのバッファリングの生成（複数文字をまとめて読み込む。１行単位など）
        BufferedReader tmpBuffer = new BufferedReader(new InputStreamReader(inputStream));
        // 複数行の文字列を連結するオブジェクトの生成（String より処理が速い）
        StringBuffer sequence = new StringBuffer();
        //  1 行分の文字列を入れておく変数
        String readText;
        // バッファのデータを１行ずつ繰り返し読み出す。括弧内では代入と比較を一緒に記述
        while ((readText = tmpBuffer.readLine()) != null) {
            // StringBuffer のオブジェクト sequence では文字列を連結する append メソッドを持つ
            //  1 行のデータの最後には改行を追加する。追加しないと改行されない
            sequence.append(readText + "¥n");
        }
        // sequence オブジェクトに格納された文字列をテキスト編集ボックスに表示
        et2.setText(sequence);
        // 入力ストリームをクローズ
        inputStream.close();
    } catch (Exception e) {      // 例が処理
        // Toast にメッセージを表示
        // 通常は下記のように３行で記述するが、メソッドチェーンという手法により１行で記述
        // Toast toast = new Toast(this);
        // toast.makeText(this, "ファイル" + name + "が見つかりません", Toast.LENGTH_LONG);
        // toast.show();
        Toast.makeText(this, "ファイルが見つかりません", Toast.LENGTH_LONG).show();
    }
}
```

図16　デザインパターンを意識しない場合の解説

```
if (v == bt[0] ) {   // Read ボタンがタッチされたとき

    // ファイルからデータを読み出すときは try と catch で囲まないとエラーになる
    try {

        // 変数 name に入っている名前のファイルからテキストデータを読み出す
        FileInputStream inputStream = openFileInput(name);
        BufferedReader tmpBuffer = new BufferedReader(new InputStreamReader(inputStream));
        StringBuffer sequence = new StringBuffer();
        String readText;
        while ((readText = tmpBuffer.readLine()) != null) {
            sequence.append(readText + "¥n");
        }

        // ファイルから読み出した内容を et2 にセット
        et2.setText(sequence);
        inputStream.close();

    } catch (Exception e) {
        Toast.makeText(this, "ファイルが見つかりません", Toast.LENGTH_LONG).show();
    }

}
```

図17　デザインパターンを構成するパーツを意識した場合の解説

意識せずに，各行がどのような機能を実現しているかを先頭に“//”を付けてコメントとして細かく解説している。これに対して，図17はデザインパターンとしてあとで利用されることを想定しているので，各行がどのような機能を実現しているかという詳細を理解することは必要でないため，パーツという機能のまとまり毎に簡単にコメントでその機能を説明している。図16は説明が細かく一見親切のように見えるが，デザインパターンを構成するパーツという単位がわからなくなっており，かえって理解しにくいものとなっていることがわかる。

2012年度から2014年度まで筆者の２年生のゼミの学生に，アプリ開発の教育を行ってきた。2012年度と2013年度では，学生が興味を持つだろうという思いでキャラクターがブロックをジャンプしながら歩き進むというゲームタイプのデザインパターンも教育したが，３年生になって各自アプリを企画する際には，それをデザインパターンとして活用するには難易度が高いということがわかり，表３にある2014年度の２年生用のデザインパターンからは除外している。デザインパターンの教育で重要なことは，流用が容易なデザインパターンの準備，わかりやす解説，デザインパターンを実際に組み合わせる訓練である。

5.　ゼミの学生によるアプリ開発の事例

スマートデバイス用アプリ開発の教育に，ソフトウエア開発におけるアジャイルおよびデザインパターンという考え方を導入した。このことによりどのような効果があったのかを考えるために，2014年度末の段階で実際の３年生および４年生がどのようなアプリを作成できているかを述べる。なお，前述の表１に示すように，２年生でアプリ開発の基本を教育し，３年生からの卒業研究の中で各自で企画したアプリの開発までを行い，それぞれが自分のアプリについて卒業論文としてまとめるというかたちで進める。３年生の最初に，各学生にアプリの企画を考えてもらう際には，ゲームを題材とせずに，教育的なアプリや実用的なアプリを企画するように指導している。さらに，すでに Google Play のマーケットに存在するアプリと同じようなアプリではなく，既存のアプリを十分調査して上で，既存のアプリにはない機能を含むアプリを企画するように説明を行っている。

5.1　４年生の事例

現在の４年生のゼミの学生は９名で，それぞれが３年生の最初の段階で自ら作成するアプリを企画し，３年生の最初からアプリ開発を始めた。表４は，それぞれの学生が企画したアプリのテーマと2014年12月20日時点での開発状況をまとめたものである。３年生からは卒業研究となるので，自分が企画したアプリを開発する過程でわからないことがあった場合は，最初はそれぞれの学生が書籍やインターネットで調べながら独自で問題を解決して進むことの重要性を示した。そこで解決できない場合は筆者が新たなデザインパターンを示すなどの対応を行った。ただ，この学年は，筆者がゼミの学生にアプリ開発の教育を行った最初の年であり，２年生の段階では，例題アプリにおいて学生が興味を示すだろうと考えたゲームを題材としたり，ソースコードの各行の意味を解説したりしたため，例題としたデザインパターンの難易度が高く，その数も多くなかった。

こうした理由もあり，デザインパターンの組み合わせでアプリを開発するという考え方を十分伝えきれておらず，４年生の卒業論文作成を開始する時点でアプリを完成することができない学生も複数名いる結果となった。週１コマ90分という限られた時間ではあるが，全員がアプリを完成まで支援して行くように工夫することが課題であると痛感した。ただ，アプリを完成

表 4　4 年生のゼミの学生のアプリ開発の内容と状況（2014年12月20日時点）

学　生	アプリのテーマ	開　発　状　況
学生4-01	インストールアプリの管理	時間的な面で完成には至らず
学生4-02	テニススコア記録表（団体戦）	Google Play への登録完了
学生4-03	デザインを工夫した電卓	時間的な面で完成には至らず
学生4-04	スケジュール管理	機能限定ながら完成
学生4-05	スクールバス時刻表	Goolge Play への登録完了
学生4-06	脳トレパスル，パスワード作成	脳トレパスルは Google Play への登録完了 パスワード作成は一部制約があるが完了
学生4-07	質問に基づくお勧め紹介	機能的に完了
学生4-08	期限付き自動ファイル管理	難易度の高さから完成には至らず
学生4-09	Q&A 作成支援	ほぼ完了

図18　Google Play に公開されている 4 年生の学生による 3 つのアプリ

できなかった学生の中で，なぜ自分がアプリを完成できなかったのか，アプリ開発のどこに難しさがあるのかについて，自分の体験を卒業論文のテーマにして現在まとめている学生もいるので，そのような分析は本人に取っても筆者に取っても非常に貴重であり，その論文のでき上がりが待たれる。

一方，3 名の学生は，自分で企画したアプリを完成させて，それを Google Play のマーケットに公開している。図18は，アプリを完成して Google　Play で公開しているアプリの Google Play でのアプリ説明の一部を示したものである[37－39]。これら 3 つのアプリは Google Play への登録時期も想定する利用者も異なるが，それぞれ実際にダウンロードされている。

「ソフトテニス　スコア記録表」というアプリの場合は，アプリ画面や Google Play での説明は英語圏では英語表示になる多言語化対応を

行っていることもあり，現在のダウンロード数は100～500と Google Play に表示されている。また，この学生は，このアプリをベースに「バレーボール　スコア記録表」というアプリも Google Play に公開しているが，この学生の卒業論文[40] によると図19のように，海外でもダウンロードされている様子である。このアプリの例のように，日本国内に住みながらも全世界の利用者を対象とすることできるため，アプリ開発という情報技術の面だけでなく，ビジネスの面での教育効果も期待できる。このアプリを開発した学生は，卒業論文の「おわりに」の中で，「私自身，ゼミナール活動でアプリの開発を行い，苦労したことも多かったが完成したアプリがプログラミング通りに動くことに感動した。また，Google Play Store に公開をして利用者の数などを知った際には「こんなにも自分の開発したアプリが利用されているのか」と喜んだ。公開している以上，これからもユーザーは増加していくはずである。これからはユーザーが満足できるアプリ開発を行っていきたい。」と締めくくっている。

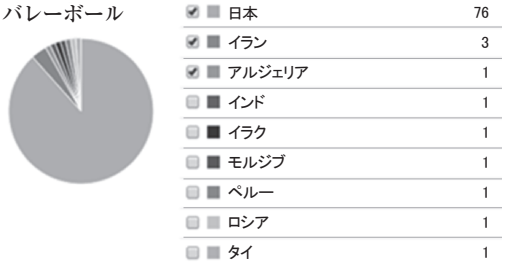


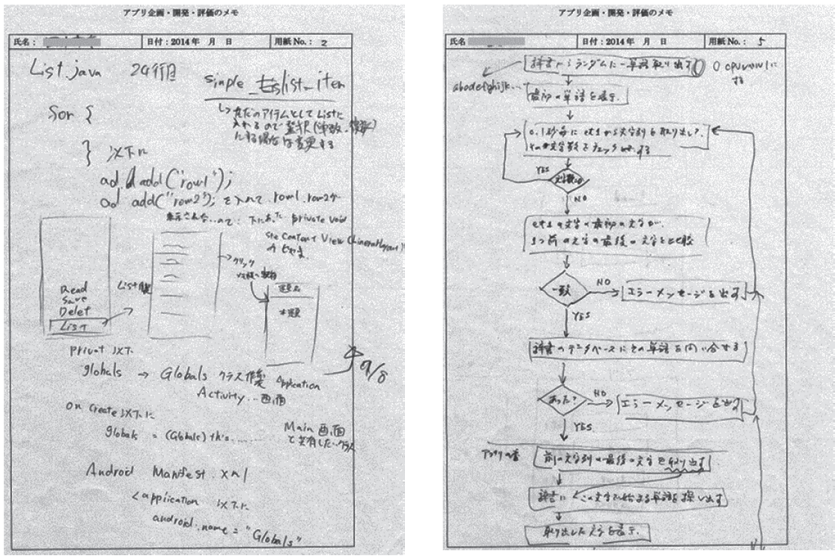
図19　海外でのダウンロード数の内訳

5.2　3年生の事例

現在の３年生のゼミの学生は12名で，それぞれが３年生の最初の段階で自ら作成するアプリを企画し，アプリ開発を始めている。表５は，それぞれの学生が企画したアプリのテーマと2014年12月20日時点での開発状況をまとめたものである。この学年では，自らアプリを企画してもらったあと，２年生で学んだデザインパターンの組み合わせで実現できるかを筆者が十分確認し，実現が難しいと思われる企画については，その旨を学生に伝えてより実現が容易なアプリを新たに企画してもらうという調整を行った。その結果，現時点でそれぞれの学生の進捗度合は異なるものの，アプリ開発の断念ということには至らずに，少しずつ完成に向かっ

表５　３年生のゼミの学生のアプリ開発の内容と状況（2014年12月20日時点）

学　生	アプリのテーマ	開　発　状　況
学生3-01	写真の分類支援	基本的な分類操作機能はほぼ完了
学生3-02	英語のしりとり	チェック機能なしのしりとり機能はほぼ完了
学生3-03	イラストによるしりとり	基本的なしりとり機能はほぼ完了
学生3-04	トランプ操作	トランプのテーブルへの配置部分まで完了
学生3-05	サイコロとコイントス	最終的な機能が完了し，新たな機能を追加中
学生3-06	カレンダー連動の家計管理	カレンダー表示とその上への金額表示を作成中
学生3-07	Web 表示のための色彩調和	田の字の４ヶ所への色指定機能を完了
学生3-08	グループ分け支援	人数とグループ数を指定したグループ分け完了
学生3-09	時間割作成管理	時間割表示画面と入力機能まで完了
学生3-10	脳活性化パスル（神経衰弱）	基本操作まで完了
学生3-11	運試しパスル	ステージ１を完了し，ステージ２を作成中
学生3-12	画像・音声を含むメモ管理	テキストメモ管理を完成し，写真メモを作成中



a）学生が自分でメモした部分　　b）筆者が学生に説明した部分

図20　各ゼミの学生のアプリ開発のためのメモの一部

ている。

週１コマ90分という限られた時間で12名の学生のアプリ開発に対応して行くために，ゼミの時間の中で機能を実現するソースコードの記述について質問した学生には，簡単に対応できる場合にはその場で対応を行い，新しいデザインパターンを示す必要がある場合には，筆者が次の週までそのデザインパターンを準備して，次の週のゼミにおいてデザインパターンのソースコードを説明して組み入れてもらうようにしている。この方法により，１コマ90分の限られた時間の中で複数の学生が抱えている疑問を解決して，先に進んでもらうことができる。

また，各学生は，３年生の１年間，場合によっては４年生の前期までアプリ開発を継続し，開発の中で遭遇する様々な問題を自らまたは教員に教えてもらうことにより解決して行く。このように，問題に遭遇してそれをどのように解決したのかということは，卒業論文をまとめる際には重要な内容となる。そのために，毎週のゼミの最初に各学生用のアプリ開発メモを配布し，ゼミの中で解決したことをメモしてもらい，また私が説明する際にもそのメモに書きながら説明して行くかたちにしている。そして，ゼミの時間の最後に各学生からそのメモを回収して筆者が保存するという方法を採っている。図20の a）は学生が作成しているメモの一例であり，b）は筆者が作成に学生の個別に説明した際に学生のメモの中に記入して説明した例を示している。学生に保管を任せるとゼミに持参し忘れたり，紛失してまう恐れがあるため，病院におけるカルテのようなかたち運用している。病院のカルテと異なる点は，学生自身も書き込めるという点，卒業論文作成の際には学生に返却するという点である。

6.　お　わ　り　に

本学のビジネス情報学科において，筆者のゼミではアプリ開発の教育を行ってきた。このアプリ開発は，単にプログラミング技術を高めるためだけが目的ではなく，そのアプリがどのように社会の人々に役に立つかを学生に考えてもらいながら企画し，そして開発してもらうことを目的としている。現代の IT 化社会において，

パソコンと同等以上にスマートフォンやタブレットなどのデバイスで動作するアプリの活用が重要となっており，また学生もこうしたアプリは日々利用しているために関心も高い。こうした題材を学校での教育に取り入れることは，スマートデバイス用アプリ開発の人材育成という社会的要請に応えることができると同時に，学生自身の関心の高さと積極的な取り組みにより教育的効果も高いと考えられる。

これまでゼミの学生にアプリ開発を教育してきた限られた経験の中で，留意しなければいけない３つの点を最後にまとめたいと思う。１つ目の留意点は，２年生において教育するデザインパターンでは，デザインパターンの数を多く教えたとしてもそれらを組み合わせて活用できなければ意味がないため，デザインパターンの組み合わせの訓練も十分行うということである。教えるデザインパターンの数とそれらを組み合わせる訓練のバランスを十分に検討する必要がある。

２つ目の留意点は，３年生において独自にアプリを企画する際には，最終的に Goolge Play にアプリを登録して公開することを前提として，Google がアプリ開発者に対して課している規約である「Google Play デベロッパー プログラム　ポリシー」[41] を十分に学生に説明することである。一見問題がなさそうなアプリであっても，Google によってポリシー違反と判断されるケースがあり，この場合アプリは強制的に公開が停止されてしまう。このようなポリシー違反により，Google Play へのアプリの登録がそれ以降一切できなくなることもある。たとえば，アプリ自身はまったく問題ないにもかかわらず，Google Play に登録する際にアプリの説明文が冗長すぎると，Google によってスパムアプリ（極端な営利目的のアプリ）と判断され，ポリシー違反になることもある。また，アプリに広告を入れて収益を考えたアプリでは，アプリの動作確認などの段階でも決して自分でその広告をクリックしないようにと学生に注意を喚起することも重要である。

３つ目の留意点は，３年生において自分が企画したアプリを学生が開発する際，わからないことがあったらできるだけ自らインターネットで検索して，問題を解決するように指導することである。アプリ開発に関する書籍も多数あるが，アプリ開発で問題に遭遇した際には，インターネットにはそれらを解決するのに十分な情報が公開されている。教員があまりに関与しすぎて，自分のアプリのソースコードの内容が理解できなくなるようなことをなるべく避ける必要がある。そのために，自分で調べて問題を解決することを教えることが重要であり，このことにより卒業後であっても自分でアプリ開発できるようになって行くことが望ましい。

注

1） 今井拓司，中道　理，木村雅秀，大下淳一（2013）「新旧交代，雪崩を打って」『特集 半導体 次の担い手』，日経エレクトロニクス 2013/07/08号，pp. 28–29.
2） 松浦晋也（2013）「2007年　初代 iPhone 発売 スマホ時代の幕開け」日経パソコン，2013/07/08号，pp. 74–75.
3） 大森敏行（2008）「「Android Market」登場，見えてきた Google 発ケータイ」日経エレクトロニクス，2008/09/08号，pp. 14–15.
4） 松元英樹（2008）「初代 Android 端末が登場米国で22日発売，自由な開発環境を持ち込む」日経コミュニケーション，2008/10/15号，pp. 18–20.
5） James O'Toole（2014）“Mobile apps overtake PC Internet usage in U.S.,” http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/.
6） 小林直樹（2013）「アラサー女性のパソコン離れが顕著　―スマートフォン利用者500人アンケート調査」日経デジタルマーケティング，2013/12号，pp. 12–13.
7） Sarah Perez（2014）“Mobile App Usage Increases In 2014, As Mobile Web Surfing Declines,” http://techcrunch.com/2014/04/01/mobile-app-usage-increases-in-2014-as-mobile-web-surfing-declines/.
8） 「スマホかざすと「津波が・・・」 高さ再現アプリ」日本経済新聞，2014年３月15日朝刊，p.

42.
9） 藤井勝敏，棚橋英樹（2013）「タブレット PC を用いた福祉分野支援アプリの開発（第 2 報）」岐阜県技術情報研究所研究報告 第15号，pp. 41–44.
10） 檜谷直樹，松原行宏，岡本 勝（2014）「タブレット PC とポータブルな反力デバイスを用いた滑車の仮想実験環境」先進的学習科学と工学研究会（第71回），pp. 13–16.
11） 木谷 篤，中谷多哉子（2014）「タブレット端末で利用される手書きノートアプリケーションのための新しいメニューデザイン」夏のプログラミング・シンポジウム2014.
12）「オープンデータ戦略の推進」http://www.soumu.go.jp/menu_seisaku/ictseisaku/ictriyou/opendata/index.html.
13）「オープンデータ生かせ 自治体，情報公開相次ぐ」日本経済新聞，2014年 3 月17日朝刊，p. 37.
14）「オープンデータ・アプリコンテスト」http://www.opendata.gr.jp/2013contest/.
15）「アプリ開発教材 学校に無料提供」日本経済新聞，2014年 8 月14日朝刊，p. 13.
16）「小学生を対象にアプリ作り教室」日本経済新聞，2014年 6 月26日朝刊，p. 13.
17）「サイバーエージェント 小学生にアプリ開発指南」日本経済新聞，2013年 9 月25日夕刊，p. 3.
18）「アプリ開発 競う10代」日本経済新聞，2014年10月31日夕刊，p. 9.
19）「プログラミング 創意工夫を育む」日本経済新聞，2014年 4 月 4 日朝刊，p. 9.
20） 独立行政法人 情報処理推進機構 IT 人材育成本部（2014）「IT 人材白書 2014」.
21）「GIFU・スマートフォン・プロジェクト」http://www.pref.gifu.lg.jp/soshiki/shoko-rodo/joho-sangyo/index.data/smartphonepj1-7.pdf.
22）「青森県における新産業創造への挑戦」http://www.pref.aomori.lg.jp/soshiki/shoko/sozoka/sozo.pdf.
23）「飯塚市 e-ZUKA スマートフォンアプリコンテスト」http://www.city.iizuka.lg.jp/03sangyou/koyo_jinzai/jinzaiikusei/smart_phone/.
24） 早藤素史（2013）「スマートフォンのアプリ開発の授業について」http://www.jikkyo.co.jp/download/detail/61/9992656090.
25） 廣田大地（2014）「App Inventor を用いた語学学習アプリ開発方法及びその演習への導入について」教育システム情報学会研究報告 28(5)，pp. 33–36.
26） 五月女仁子（2014）「App Inventor を使ったプログラミング教育の試み」日本情報科教育学会第 2 回研究会.
27） 岡崎哲夫，本郷節之，稲垣 潤（2013）「Android スマートフォンを用いたアプリケーション演習教材の開発」工学教育研究講演会講演論文集（61），pp. 94–95.
28） 吉永耕介，安部真人（2014）「SPI テスト トレーニングのアンドロイドアプリの試作」神田外語大学紀要，巻26，pp. 217–240.
29） Derek Riley（2014）“Using Mobile Phone Programming to Teach Java and Advanced Programming to Compter Scientists,” Proceedings of the 43rd ACM technical symposium on Computer Science Education, pp. 541–546.
30）「動かないコンピュータ」日経コンピュータ，2014/11/27号，pp. 144–145.
31） Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, “Manifesto for Agile Software Development,” http://agilemanifesto.org/.
32） 坂本直紀（2010）「ウォーターフォール開発と使い分け」日経コンピュータ，2010/02/03号，pp. 120–123.
33） Kent Beck, Ward Cunningham（1987）“Using Pattern Languages for Object-Oriented Programs,” Conference on Object-Oriented Programming, Systems, Languages & Applications.
34） Erich Gamma, Richard Helm, Ralph Johnson, John Vissides（1994）“Design Patterns: Elements of Reusable Object Oriented Software,” Addison-Wesley Professional.
35）「XML レイアウトと Java プログラムレイアウトについて」https://groups.google.com/forum/#!topic/android-group-japan/lbSVkdcK0Xc.
36） “Best practices: Layouts on Android（Programmatic vs XML）” http://stackoverflow.com/questions/9827819/best-practices-layouts-on-android-programmatic-vs-xml.
37）「パネルクイズ」https://play.google.com/store/apps/details?id=jp.co.jz4o.quiz1&hl=ja.
38）「ソフトテニス スコア記録表（団体戦）」https://play.google.com/store/apps/details?id=jp.co.w99k.softtennis&hl=ja.
39）「広島経済大学スクールバス時刻表」https://play.google.com/store/apps/details?id=jp.co.srch.huebus&hl=ja.
40） 脇 裕弥（2014）「スマホアプリ開発と Google Play Store 公開におけるインストールの分析」広島経済大学卒業論文.
41）「Google Play デベロッパー プログラム ポリシー」https://play.google.com/intl/ALL_jp/about/developer-content-policy.html.