N-015

スマートフォンアプリケーション作成の教育法について
Education of Smartphone Application Development

小林　洋†
Hiromi Kobayashi

１．はじめに

スマートフォンが学生にとっても日用品化している現在，スマートフォンのアプリケーション（以下，スマホアプリ）を作ることに興味を持っている学生は多い．しかし，スマホアプリは，イベント駆動型のプログラムのため理解するには初心者には難易度が高く，従来のプログラミングの授業のように，数十人～100 人程度を対象とするような大人数教育では，入門用のサンプルプログラムの小修正程度で授業が終わってしまう可能性が高い．また，その開発支援環境の設定がアプリ開発の一部と見なせる程密接な関係がある上に，開発環境はしばしばバージョンアップされるので，授業においては，単にプログラミングを教えるだけではなく，開発支援環境のインストールや設定から教えた方が望ましいように思われる．すると，どうしても高学年で 10 人程度の少人数で行われるゼミナール等で扱うしかない．但し，教える内容は，きっかけを与えるための入門的なもので良いと思われる．その後は，インターネットでの解説記事や参考図書で容易に学ぶことが出来るためである．学生がアプリを作成しようとする場合，インターネットで参考となるサンプルを捜すのは止むを得ないと考えるが，サンプルから，自分の作ろうとするプログラムに修正，拡張する力をつけるためには，先ずは初歩のプログラミング入門の授業から，出来ない場合に安易に解答を教えるのではなく，考え方やデバッグの方法やエラーが出た時の対処の仕方が身に付くような教育方法が望ましいと考えられる．しかし，プログラミングの入門の授業においては，数十人から 100 人規模のプログラミングの実習授業が普通であり，学生に考えさせるような授業，つまりアクティブラーニング的な授業をいかに行うかというのは課題である．

２．背景と問題点

スマートフォンが日用品，更には生活必需品のようになって来た現在，学生にとっても日常生活の中で良く使うものであるせいか，スマホアプリをちょっと作ってみたいという希望は，情報系の学科に入学してくる学生には多いように思われる．また，卒業研究においても，ソフトウェア開発を看板に掲げているような研究室を希望するような学生には，スマホアプリの作成，特に，ゲームの作成や Twitter 等のデータを使ったアプリの作成の希望が少なくない．そこで，スマホアプリの作成について，授業として取り上げられないか検討すると，次のような問題に直面する．

（１）プログラミングの難しさ

スマートフォンのプログラムは，情報系学科でのプログラムの入門教育で通常良く使われるＣ言語などでのプログラムのように，プログラムに書いた一連の処理を実行し実行が終わると終了するというフロー駆動型の振る舞いをするのではなく，プログラムは通常待機状態にありユーザの操作に対応した処理が行われるというイベント駆動型の振る舞いをする．イベント駆動型のプログラミングは，フロー駆動型のものに比べ，一般的に複雑で，しかもプログラマから見るとブラックボックスの部分が大きく難度が高い．この事は，通常 Java で作る Android のアプリであろうと，Objective-C で作る iOS のアプリであろうと変わらない．普及率の高い Android については，アプリは標準的には Java 言語で作られるため，Java は最近の情報系の学生はほとんどが履修しているであろうから，Android でのプログラミングは学生にとって習得が容易であると，かつては言われていた事もあった．しかし，Android で使われている Java には独特の部分があり，GUI についても Java では良く用いられる Swing ライブラリも用いていないので，Java だから習得が容易という事にはならない．

また，最近のプログラミングでは，Android に限らず，フレームワークを用いたりライブラリを多用する傾向があるため，プログラミングの際には 1 行ずつ理解しながらコーディングして行くというよりは，「このような場合にはこのような枠組みやパターンを用いる」と現象論のようにとらえ，それを修正・拡張して行くのが，作成者にとっては普通になって来ているように思われる．

（２）開発環境や Android のバージョンアップと環境設定

スマホアプリのプログラミングにおいては，開発支援環境の設定がアプリ開発の一部と見なせる程密接な関係がある上に，開発環境はしばしばバージョンアップされるので，単にプログラミングを教えるだけではなく，開発支援環境のインストールや設定から教えた方が望ましいように思われる．すると，教育としては 10 人程度の少人数で行うゼミナール等で行わざるを得ないように思われる．開発環境については，ここ数年，大きな変化があった．従来は開発支援環境 Eclipse[1]に ADT(Android Development Tool)[2]をプラグインとしてインストールして用い，Eclipse の日本語化版については Pleiades のサイト[3]のものを使うのが一般的であり，インストールの方法は何度か変更があった．ところが，2014 年末からは，Android Studio[2]を用いるのが標準となった．また，この他に，Android のバージョンごとに Android SDK をインストールして用いる必要がある．更に，PC 上での Android 端末のエミュレータとしては，AVD(Android Virtual Device)が Android SDK に付属しているが，最近では，より高速の Genymotion[4]等を用いるようになって来ており，開発支援環境が大きく変化している．

† 東海大学情報通信学部，Tokai University

３．インターネットの活用と小規模アプリの開発

（１）サンプルプログラムと不具合対処記事の存在

現在，インターネット上には Android アプリはもちろんの事，様々なサンプルプログラムが存在し参考図書も多いためか，卒業研究などでプログラムを作成する場合には，まず，自分で考えるよりも参考になるプログラムを捜すことから始めるケースが多い．但し，巧みに部品化が行われていたり，デザインパターン[5]を用いたようなものがあると，理解するのに非常に時間がかかる．

また，インターネットには、様々な不具合対処方法の記事が存在しており、プログラミング中にエラーが出た場合には，考えるよりも検索した方が，早く解決できる事がある．これについては，学生に積極的に勧めても良いように思われる．

（２）小規模アプリの開発方法

スマホに限らず，アプリケーションの開発において，まだ開発に慣れていない学生に設計図を書かせると，かなり自由な発想で設計図を書いていく傾向がある．ところが，プログラミングの段階になると，自分の書いた設計図どおりでは作るのが困難となり，一旦，設計図は棚上げにしてボトムアップでプログラミングを行うことになり，設計図とは違ったものが出来てしまうことを良く目にする．但し，スマホアプリに関しては，日常的に使っているせいか，画面設計についてはあまり無理のないものを作るようであり，小規模のスマホアプリ作成なら，設計図としては，画面設計のスケッチ程度[6]で良いように思われる．

４．教え方

通常の授業科目として，数十人から 100 人程度を対象に Android アプリの作成について教えようとするならば，開発環境のインストールや設定から教えたいところだが，これは不可能に近い．また，Android アプリは，イベント駆動型のプログラムであり，比較的難易度が高いので，多くの学生を対象に，ある程度理解させながら実習をするのは困難が伴う．そのため，プログラミングに強く興味を持つような学生が 10 人程度集まる高学年でのゼミナールや卒業研究の前段階で，扱うのが妥当と考えられる．

Android のアプリ作成で学生に教える内容は，ごく入門的なもので良いだろう．その理由は，インターネットでの参考記述やサンプルプログラムが大量にあり，多くの参考図書が出版されている現在では，高学年であれば，きっかけを与えるための入門的な教育を行えば，後は学生自らが，必要性があれば学んでいくことが容易なためである．Android の構造（アーキテクチャ）に沿ったアプリの作成のための仕組み，例えば，画面遷移を行うためのインテント（intent）や画面を持たずにバックグラウンドで行う処理であるサービス（service），あるアプリが他のアプリのデータやデータベースを用いる場合に必要となるコンテントプロバイダ（content provider）等の仕組みについては，先ずは仕組みについて教えるのみに留め，プログラミングについては，後で具体的に必要になった際に習得してもらえば良いのではないかと思われる．

５．おわりに

スマホアプリの作成の教育と，それ以前のプログラミング入門での教育について，次のような事を提言したい．

（１）スマホアプリでは，開発支援環境の設定がアプリ開発の一部と見なせる程密接な関係があるので，開発支援環境のインストールや設定から教えるのが望ましい．

また，スマホアプリでの，イベント駆動型プログラミングは難易度が高い．従って，授業は，10 人程度の少人数の学生を対象に，高学年で教えるのが現実的であろうと思われる．

（２）スマホアプリの設計については，画面のスケッチ程度で行わせるのが良いように思われる．

（３）スマホアプリに限った事ではないが，少し大きなプログラムの作成においては，枠組みや中心となる部分から段階的に徐々に作って行くことや，部品化することも考慮する事なども教える必要がある．

（４）スマホアプリ以前の大学初年度のプログラミングの入門段階から，次のようなアクティブラーニング的な教育を行うのが望ましいと考えられる．

・学生は，プログラムが出来ないとＴＡや教員に解答を直ぐに聞きたがる傾向があるが，解答を教えるより，考え方やヒントやデバッグの方法を教えた方が良い．

・デバッグの方法として，出力文を途中に入れて途中の値をチェックする等のハウツー的な方法を多少なりとも教えた方が良い．Android での開発においては，デバッグ用の Logcat を用いる事を早い内に教えておいた方が良い．

（５）学生のインターネット活用は時代の流れなので，特に高学年においては，ソフトウェア作成においてサンプルプログラムを捜したり，エラーメッセージの解析に活用するのは，認めても良いと思われる．但し，インターネット上に掲載されているプログラムは，開発支援環境の設定や，バージョン変更に伴うプログラムの修正等を行わないと，そのままでは動かない場合が珍しくないので，これについては注意点として言っておく必要がある．

インターネットにサンプルやエラーについての参考記述が多数掲載されている今日では，それを教育に積極的に活用する事を考える必要があるだろう．しかし，一方では，安易にサンプルが手に入るために，プログラミングについて，考えるよりも解答を捜すことに陥り易いので，これを克服するような授業方法については課題である．

参考文献

[1]Eclipse Foundation: https://eclipse.org/
[2]Android Developers, Android Studio: http://developer.android.com/intl/ja/sdk/index.html
[3]MergeDoc Project Pleiades: http://mergedoc.osdn.jp/
[4]Genymotion: https://www.genymotion.com/
[5]E.Gamma,R.Helm,R.Johnson,J.Vlissides 著，本位田真一，吉田和樹監訳：オブジェクト指向における再利用のためのデザインパターン，ソフトバンク(1995)
[6]深津 貴之,荻野 博：プロトタイピング実践ガイドスマホアプリの効率的なデザイン手法, インプレス(2014)