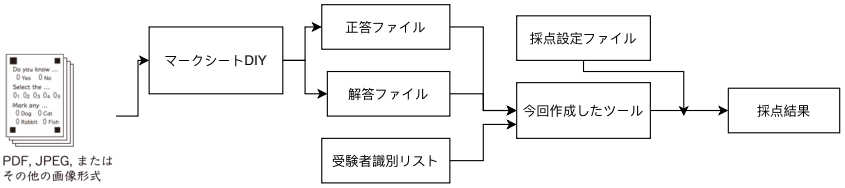
小規模マークシートテストの自動採点ツールの提案

学籍番号 644439　ネットワークメディア学科　木本　涼（荻原研究室）

1 背景

現在、多様なマークシート自動採点ツールがあり、そのサービスもツールによって異なる。しかし、一貫して大規模で組織的なマークシートテストを想定したツールであり、学校の小テストといった小規模なテストでは複雑で扱い難い。そこで今回、小規模なテストに最適な自動採点ツールの提案を行う。



概要図

2 目的

採点するにあたって、テストの正解と受験者の解答を読み取る必要がある。他社製品[1][2]では専用のマークシートや専用の読み取り機器が必要なものがあり、小テストの採点に運用するにはコストがかかりすぎる。そこで、普通紙とスキャナがあれば小テストを実施できる「マークシートDIY」というツールを使用する[3]。しかしこのツールにはデータ化する機能のみで採点する機能は実装されていない。そこでこの「マークシートDIY」の手軽さを生かしたツールを作成する。また、学校で使用することを想定した場合、同じテストをクラスごとに採点することが考えられる。そこで何度も同じ採点設定をしなくても良い様にするツールを作成する。この二点を目標に実装を行う。

3 作成したツールの説明

作成したツールは、三つのファイルが必要である。一つ目は正解が記載された正答ファイルである。二つ目に受験者の解答が記載された解答ファイルである。この二つのファイルは「マークシートDIY」の出力から直接入力できるよう、TSV形式とした。三つ目のファイルは受験者の識別番号のリストファイルである。これは欠席者が一目でわかるようにするために実装した。

採点の設定は、設定を行う問題範囲を入力させ、その範囲での採点方式の選択、配点の入力という単純なものにした。また、全ての採点設定が終えた時、次回でもその採点設定が使用できるよう、採点設定ファイルに保存するようなっている。このファイルはJSONで記載されている。結果は受験番号と該当する得点のみが表示される。

4 評価

実装した採点方式と配点で正確な点数のみを表示させることができ、他社製品より簡易的でかつ、わかりやすいツールであると言える。操作性に関しては採点方式のルールを理解しなければならない点以外、ファイルを入力するだけで良いためユーザの負担は少ない。しかし得点のみの出力であるため、詳細なデータを表示させたい場合やデータ分析などの操作には適していない。このことから、このツールは復習をしないテスト、例えば、受験者の理解度を測るといった用途で使用するのが最良であると考えられる。

5 課題

使用者によっては平均点を出力して欲しい人がいたり、復習に使用するために、個別にどこを間違えたのか分かるようにして欲しい人がいたりする場合が考えられる。そのために、使用者で自分の使いやすいように拡張できるツールにすることが求められると考える。しかし、この拡張もサービスを多くしすぎてしまうと、設定自体のルールを理解してもらう必要があるため、ユーザの負担は増加してしまう。この設定量とユーザの負担のバランスを考慮したツールを作成することが今後の課題である。

参考文献

[1]スキャネット-らく点マーク君３紹介ページ
https://www.scanet.jp/wp/?p=3362

[2]マークシート導入で作業効率化のススメ
http://www.convenient-smooth.net/ranking/d_sol.html

[3]マークシートDIYの使い方
http://www7a.biglobe.ne.jp/~ogihara/ja/Manual_0.7.1ja.pdf

iOS アプリにおけるプレゼンテーションロジックと状態管理に注目したアーキテクチャの提案

学生証番号 645177　　インテリジェントシステム学科　　松山 友也 （荻原研究室）

1 はじめに

近年の iOS アプリでは、様々な要因により、アプリケーションは複雑化、大規模化している。そのような大規模なアプリケーションにおいて、既存の開発手法で開発すると、クラスが肥大化していた。さらに、アプリケーション全体で多くの状態を網羅的に管理する難しさが課題であった。そこで、これらの課題を解決できるアーキテクチャを提案し、実装した。また、評価としてサンプルアプリを開発した。結果として、既存の開発手法より状態管理が簡単になり、明示的でわかりやすいシステムを開発できたと考える。

2 Redux

Redux は JavaScript で提案されているアーキテクチャである[1]。多くの変化し続ける状態の管理に注目したアーキテクチャである。また、単一方向のデータフローを厳守するように考えられている。

3 提案するアーキテクチャ

はじめに、従来の開発で用いられるアーキテクチャは、Cocoa MVC が多い [2]。また、画面を作成するためには UIViewController という Apple 提供のクラスを使う必要がある。このクラスでは、View と Model を管理・保持している。

Cocoa MVC での開発では、システムの大規模化に伴い、UIViewController クラスの肥大化と、アプリケーション全体での状態管理の煩雑化が課題として考えられる。

そこで、アプリケーションにおける状態に注目し、View + ViewStream + Redux で開発できるアーキテクチャを提案する。ViewStream では、View との入出力を網羅的に定義している。

図 1 では提案するアーキテクチャの全体図を載せる。

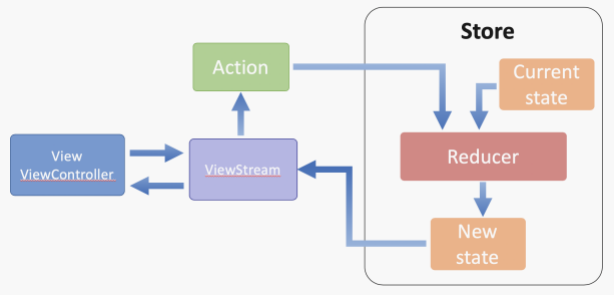


図 1 提案するアーキテクチャの全体図

4 サンプルアプリケーションの実装

画面数は 3 画面で、GithubAPI を用いて、ログイン、リポジトリの取得、star しているリポジトリの取得をするサンプルアプリケーションを実装した。

5 議論

ViewStream を作成することで、ViewController からビジネスロジックとプレゼンテーションロジックを分離でき責務を明確にすることができた。また、Protocol を活用することで、入出力の明確化ができ、テスト容易性も向上した。また、Action を用いて純粋関数でのみ状態を変更できるように制約をつけたため、状態の変化することに対して、予測可能となった。一方で、ファイル数が MVC に比べて増え、Protocol の束縛のため記述量が増えた。

5 まとめ

本研究では、ViewStream と Redux を用いたアーキテクチャの実装、それらを適応させたサンプルアプリの実装まで行った。ViewController から責務を分離することで、責務を明確にでき、ViewController の肥大化を防ぐことができた。

参考文献

[1] Redux: https://redux.js.org/introduction/motivation
[2] Apple Inc “Apple Developer Document”, https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html#//apple_ref/doc/uid/TP40008195-CH32-SW1

ユーザAPIのない監視システムから情報を取得して活用する手法の提案

学生証番号 445115　コンピュータサイエンス学科　馬場 磨美　（荻原研究室）

1 はじめに

ネット環境やIoT機器の発達に伴い，監視システムや見守りサービスへの関心が高まっている[1][2]．しかし，既存の監視システムなどには，観測データや警告を画面に表示するだけのものや，特定の企業が提供するシステム間でしか情報の共有ができないものが多く存在する．いくつかの見守りシステムや監視システムについて調査した結果，ユーザがデータを活用できるようなAPIを提供するものはほとんどないことが分かった．

ネット上の監視システムなどから提供されるデータをユーザのローカル環境のシステムと連携できれば，災害への備え，商品の仕入れの計画，農作業の支援など，さまざまな用途に活用できると考えられる．

2 手法の提案

本研究では，既存の監視システムの情報を活用する方法として，画面表示を画像として読み取って解釈し，自らが作成したローカル環境のプログラムを制御する手法を提案する．

今回は文字表示を画像として読み取り，そこから文字情報を取得してローカル環境で利用できるかどうかを検証する．

3 実験

画像の文字認識(OCR)を行うために，フリーソフトとして公開されている Tesseract（テッセラクト）[3]というパッケージを利用した．TesseractはLinuxやMacなど，幅広い環境で動作する．その特徴を活かすため，このツールとシェルスクリプトを組み合わせて確認用のプログラムを作成した．

実験では，見守りシステムなどの検出した情報がパソコン上に文字として表示されている状況を想定する．画像内の数値データを読み取り，それが基準値より大きく上回ったり下回ったりした際に警告するものを作った．

4 評価

今回使用したツールTesseractでは，文字が小さすぎると読み込めない場合があった．そのため，あらかじめ画像を取り出したいウェブページなどを拡大表示しておく必要がある．日本語の認識も可能だが，精度は高くない．

今回の条件では，一枚の画像内の情報しか取得できない．複数の情報を同時に得るには，画面上の複数箇所を切り出してそれぞれに対して文字認識を行う必要がある．

5 まとめ

見守りシステムや監視システムなどの既存のシステムの自動化を目指して，新たなシステムを試作した．

実験システムでは文字認識が可能な画像しか処理できないが，今後は画像の差分情報を利用したり，機械学習の手法を取り入れたりして利用できる対象を広げる必要があるだろう．

参考文献

[1] 菅原真司: “IoTにおけるエッジ／フォグコンピューティング技術”，電子情報通信学会誌, Vol.102, No.5, pp.413–417 (2019).

[2] 情報処理学会: “《特集》社会を変えるIoT”, 情報処理, Vol.60, No.2 (2019).

[3] Ray Smith, et al.: Tesseract, https://github.com/tesseract-ocr/tesseract