関数型言語による Podcast ダウンローダの実装について

田中　陽子*，岩見　宗弘
（島根大学総合理工学部）

1. はじめに

近年，インターネット上では，個人の日記や団体の活動記録などを書くブログが流行している．それは文字表現に止まらず音声・動画データとしても公開されている．幾つかの関連する音声・動画データ（エピソードとも呼ばれる）をインターネット上に Podcast 番組として公開する方法は，Podcasting と呼ばれており，それを行う Podcaster とその視聴者は年々増加傾向にある．視聴者が閲覧する番組の管理や，エピソードのダウンロードなどに利用する Podcast ダウンローダも数多く開発されている．関数型言語 Haskell により開発された hpodder もその一つである．しかし，hpodder は CUI(Characterbased User Interface)上の操作しか行えず，コマンドと Haskell 関数の操作方法を知らなければ，操作が非常に困難なものとなっている．また，番組の管理やエピソードのダウンロードにおいても，特定したエピソードのみのダウンロードを行うこと，幾つかの番組を指定しての更新を行うことなど，機能として存在するべきであると考えられる要素が，幾つか不足しているといった点もみられる．従って本研究では，hpodder が一般の利用者にとって，より容易に操作可能になるようにするために，GUI(Graphical User Interface)化を行い，また新たな機能を付加することで hpodder の改善を行うことを目的とする．

2. 関数型言語 Haskell

Podcast ダウンローダは，Web サーバとデータのやりとりを行うことが多いプログラムである．また実際の番組は，RSS（RDF Site Summary）と呼ばれる，サイトの更新情報が簡単にまとめられている文書フォーマットで公開されている．関数型言語は，Haskell，OCaml，ML，Lisp などのように関数を中心として，プログラムを組み立てることを目的とする(1)．また，引数に関数をとることができる．これにより柔軟なプログラミングが可能であり，コードも完結で読みやすくなる．特に Haskell は，コンパイル時の型検査が特に強力であり，引数は必要になった時だけ評価を行う遅延評価を基本としている．遅延評価を行うことにより，計算や関数の受け渡しに必要のない引数は省かれ，計算式内の同一引数は簡約化され，複数回実行されることはない(2)． Podcast ダウンローダを作成した場合，サーバとのデータのやりとり，RSS の構文解析を行う上で，無駄な時間を限りなく省くことが Haskell では可能であると考えられる．上記の理由から本研究では関数型言語 Haskell を使用する．

3. 先行研究

Podcast ダウンローダの先行研究として，上記で述べたように hpodder が Goerzen により開発されている．hpodder は OS に依存せず，実行環境を整えれば，Linux だけではなく Windows や MacOS 等でも動作する(3). マルチスレッドダウンロード機能を含めて，簡単に多くの操作を行うことができ，また他のプログラムから過去の処理結果の履歴などを簡単にインポートすることができる．さらに，簡単なカスタマイズと迅速なセットアップで使用することができる．しかしながら，hpodder は CUI 上の操作しか行えず，コマンドと Haskell 関数の操作方法を知らなければ，操作が非常に困難である．hpodder の主な通信の流れは，まずユーザが Podcast ダウンローダに対して，番組の登録，更新，削除といった操作を行う．入力により受けた命令をメインプログラムが受け取り，それぞれの処理関数へ移る．処理関数の内容によりプログラムは WEB サーバ，データベース，RSS ドキュメントを適時呼び出し，処理を実行した結果をユーザに再度返す．

4. hpodder の機能追加

hpodder プログラムの主な構成要素は，幾つかの Haskell ファイルと，プログラム中で使用されている HackageDB 内のパッケージ関数である．HackageDB とは Haskell パッ

ケージのリソースコレクションのことである．Haskell でのプログラムの作成に必要な関数をプログラム作成時に定義しなくても, 定義したい関数と同一機能の関数があるパッケージをインポートすれば使用することができる. コマンドを新たに定義する場合は Commands.hs 内に定義しておく必要がある．また各コマンドの実行プログラムは, Commands ディレクトリ内にそれぞれ Haskell ファイルとして保存されており，コマンドの実行時は Commands.hs から Commands ディレクトリ中の各コマンドに対応したファイルの関数が実行される．また hpodder 内のプログラムは以下の 4 つのシステムに大別して考えることができる. これらのシステムはそれに対応するプログラムが相互作用して hpodder 内のコマンドを実行している.

(1) 代数的データ型
(2) データベースシステム
(3) RSS の構文解析器
(4) ダウンロードモジュール

本研究では, 指定したエピソードのみダウンロードする機能と, エピソードグループを指定してダウンロードする機能を追加する．それぞれの機能の追加方法は，以下の通りである.

1. 新たに代数的データ型を宣言し, データベースへ専用テーブルを作成する（図１）.
2. それぞれのダウンロードモジュールを作成する（図 1）.

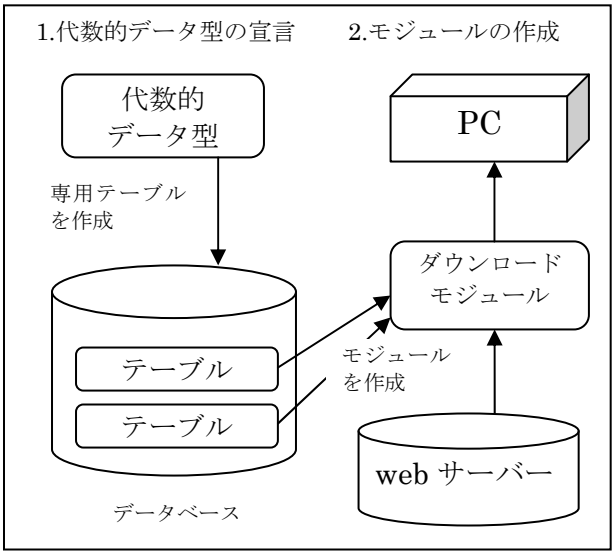

図 1　データの流れ

両機能はデータベースにそれぞれのテーブルを作成して, そのテーブルを参照できる形のダウンロードモジュールを作成している点で共通している.

4.　グラフィカルユーザインタフェース化

機能を追加した hpodder のオプションを除く GUI 化を行う．hpodder の GUI 化には，まず GUI 部品の組み立てに Glade を使用する（図 2）．Glade は図 2 のように機能を実行するためボタンを部品として作成し, この部品と実際のコードを関数を用いて結び付けるものである. これにより現在の CUI 操作を，GUI によるボタン操作に切り替えることを可能にした.

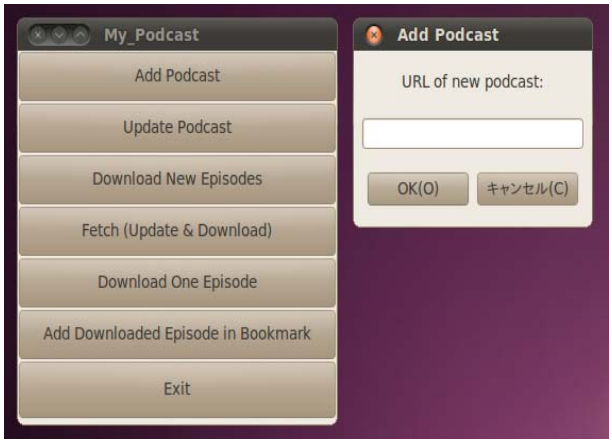

図 2　Glade による部品の組み立て

5.　むすび

本研究では，2 つの機能の追加と GUI 化を行うことで, 従来の hpodder よりも操作性を改善することができた．さらに改善できる点として番組のグループ化, 個別ユーザの登録機能などが考えられる.

文献

(1) 青木峰郎, ”ふつうの Haskell プログラミング,”ソフトバンククリエイティブ，2006.
(2) G.Hutton，”プログラミング Haskell,”オーム社，2009.
(3) B.O’Sullivan，J.Goerzen，D.Stewart, “Real World Haskell,”オライリー・ジャパン，オーム社，2009.