簡易「文芸的プログラミング」システム SWEB

Simple “literate programming” system *SWEB*

増井 俊之
Toshiyuki MASUI
シャープ株式会社
SHARP Corporation

概要

Knuth の「文芸的プログラミング」システム WEB を拡張しつつ簡易化を行なうことにより、プログラム開発のみでなく論文 / 実験レポート / テクニカルレポートなど各種の文書作成に使用できるシステム “SWEB” を作成・評価した。 SWEB はオリジナルの WEB と異なり、複数の言語に対応が可能 / 分割コンパイルに対応 / プログラム中に制御指令を入れる必要がない / 変化のないファイルは更新しない、などの特徴を持っており、 Make や RCS など既存のプログラム開発ツールと組みあわせて効果的に文書 / プログラムの開発を行なうことができる。

1 WEB システム

WEB システム [1] はスタンフォード大学の Donald Knuth が開発した、いわゆる「文芸的プログラミング」 (Literate Programming) を支援するシステムである。文芸的プログラミングとはプログラムの作成とその文書化を同時に行なう手法で、プログラムとその解説文書をひとつのファイルに混在させながら追加したり修正したりして同時に開発していくことにより、常に整合性がとれた保守しやすい文書とプログラムが作成できるというものである。

WEB システムはもともと TEX システムを記述するために開発されたものなので Pascal と TEX が対象になっているが、 C が使えるようにした CWEB システム [3]、 C と `roff` との組み合わせが使える cweb システム [5]、任意のプログラミング言語に対応させることのできる Spider[4] といったシステムも開発されている。

WEB 文書は複数のプログラムモジュールの集合という型式をとり、各モジュールは解説部分とプログラム部分とで構成される。 WEB 文書を `tangle` というプログラムに通すとプログラム部だけが抽出されてひとつの Pascal プログラムになり、 `weave` というプログラムに通すと解説部とプログラム部の両方が融合した TEX の文書になる。プログラム部は整形され、モジュール毎に章立てが行なわれ、目次や索引も完備した完全なドキュメントが生成される。このようにしてひとつの WEB ファイルからプログラムとそのドキュメントの両方が生成される。

2 WEB システムの問題点

WEB をによる文芸的プログラミングは評判の割には実際にはあまり使われていない。これは WEB に以下のようなデメリットがあるためと考えられる。

- プログラムテキストが読みにくい
- `tangle`, コンパイル, テスト, フィードバックというループの手間が大きい。
- 解説文だけでなくプログラム中にもフォーマットのための指令を入れなければならない。
- Pascal の “`:=`” が “←” と印刷されるなど読み易さのための工夫がされているが、実際のプログラムテキストと対応しないので逆効果になることもある。
- マクロ定義機能がわかりにくい
- プログラムを分割するとそれが全部モジュールとされて段落分けされてしまうなど、文書型式を自由に制御できない
- ひとつのプログラミング言語にしか使えない

  WEB は Pascal 専用であるし、 Spider を使用した場合でも複数言語ソースをひとつの WEB 文書に混在させることはできない。
- 分割コンパイルができない

  `tangle` はひとつの WEB 文書からひとつの Pascal プログラムを出力するので分割コンパイルができない。 CWEB や Spider では改善されているが、それでも分割コンパイルがし易いとはいえない。

	WEB	SWEB
応用分野	プログラム開発のみ	プログラム開発及び文書作成一般
文書整形言語	TEX に限る	任意のものを使用可
プログラム言語	Pascal に限る	任意のものを使用可
複数ファイルへの出力	不可	可
分割コンパイル	不可	可
フォーマット指定	きめ細かく可能	プログラム部は通常のプリティプリンタ並
Make への対応	非対応	対応
出力ファイルの修正	可能	禁止
仕様の大きさ	中	小
システムの大きさ	大 (`tangle`: Pascal 2700 行)	小 (`stangle`: Perl 100 行)
別ファイルの取り込み	不可	可

図 1: WEB と SWEB の比較

- `tangle` を通すたびにプログラムファイルが更新されてしまう

  `tangle` を起動する度に全プログラムファイルが更新されてしまうため、 Make をプログラム開発に使用する場合 `tangle` 起動の度に全ファイルをコンパイルし直さなければならない。
- 文書整形言語として TEX しか使えない

3 Simple WEB

WEB システムには以上のような多くの問題があるためもっと簡便な WEB が望まれる。 WEB の問題点を解決しかつプログラム開発以外の応用にも使用できるようにするため、以下のような方針を採用した Simple WEB (SWEB) システムを開発した。

- プログラム部にはフォーマット指令は入れない。プリティプリンティング技術で対処する。

  プリティプリントにはプログラミング言語毎に用意した構文情報を使う[†1]。
- プログラム部は出現順に出力する

  WEB のモジュールマクロ機能は最近のプログラミング言語では有用ではないためマクロ定義機能は設けず、プログラム部は文書内での出現順に出力することにする。
- WEB ファイルが更新されてもプログラム部分が変化しなければ `tangle` を通した後でプログラムファイルの作成日時が更新されないようにする。
- 解説部分の TEX に依存する部分を排除する

SWEB の文書ファイルの仕様を以下のように定めた。

- SWEB 文書は解説部とプログラム部から構成される。
- 制御文字列 "`@@`" のみからなる行で解説部の開始を指示する。
- 文字列 "`@@`" の後にファイル名を空白で区切って並べた行でプログラム部の開始を指示する。プログラム部は指定されファイルに格納する。ファイルが複数指定されているときは同じプログラムソースを複数のファイルに格納する。
- ファイル名の後に丸括弧 "`(`", "`)`" で囲ってプログラム言語名を指定することができる。これが無い場合はファイルの拡張子より言語を類推する。
- 文字列 "`@@i`" の後にファイル名を指定するとそのファイルを取り込む。
- 文字列 "`@@c`" の後に文字列を指定すると、以降 "`@@`" のかわりにその文字列を制御文字列として使用する。

プログラム `stangle` (Simple `tangle`) によって SWEB 文書からプログラム部を抽出することにする。またプログラム `sweave` (Simple `weave`) によって SWEB 文書を TEX 文書に変形することにする。 `sweave` は "`@@`" で段落分けをすることはせず、文書作成者が "`\section`" のような制御指令を明示的に挿入する。このため例えば関数定義の途中でも別の章を開始することができる。上述の SWEB 文書の仕様自体は TEX などの文書整形言語に依存していないので `sweave` 以外のプログラムを使うことにより TEX 以外のフォーマッタを使うことができる。

`stangle` の出力するファイルはすべて書き込み禁止に設定され、エディタなどで修正できないようになっている。修正は必ずもとの SWEB ファイルに対してのみ行なうことができるため出力されたファイルの修正により整合性が乱れることがない。

プログラムと文書だけでなくデータや図なども SWEB 文書内で扱うことにより、これら全てを一貫性をもって取り扱うことが可能になる。

WEB と SWEB の違いを図 1に示す。

[†1] UNIX では `vgrind`, `tgrind` といったプリティプリンタで使用するための `vgrindefs` という汎用のプログラム言語構文定義ファイルが用意されており、これを使用することができる。

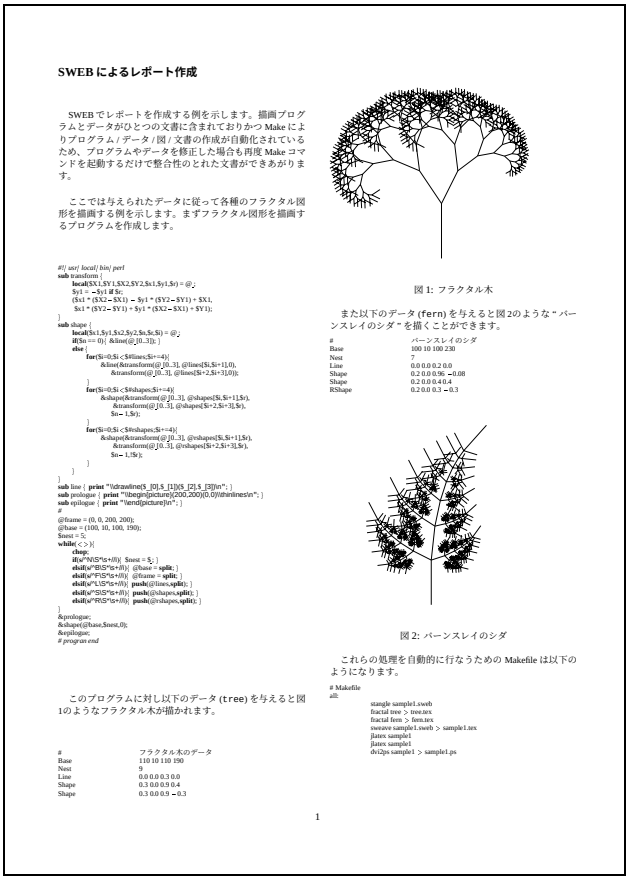
```
\documentstyle[twocolumn,times,sweb,epsf,newfloat,rcsdate,epic,eepic]{jarticle}
\setlength{\textwidth}{18cm}\setlength{\columnsep}{0.85cm}
\addtolength{\evensidemargin}{-2cm}\addtolength{\oddsidemargin}{-2cm}
\tgrindsize{\scriptsize}
\begin{document}
\small
\section*{SWEBによるレポート作成}
SWEBでレポートを作成する例を示します。
描画プログラムとデータがひとつの文書に含まれておりかつ
Makeによりプログラム/データ/図/文書の作成が自動化されている
ため、プログラムやデータを修正した場合も再度Makeコマンドを
起動するだけで整合性のとれた文書ができあがります。

ここでは与えられたデータに従って各種
のフラクタル図形を描画する例を示します。
まずフラクタル図形を描画するプログラムを作成します。

@@ fractal(perl)
#!/usr/local/bin/perl
sub transform {
        local($X1,$Y1,$X2,$Y2,$x1,$y1,$r) = @_;
        $y1 = -$y1 if $r;
        ($x1 * ($X2-$X1) - $y1 * ($Y2-$Y1) + $X1,
         $x1 * ($Y2-$Y1) + $y1 * ($X2-$X1) + $Y1);
}
sub shape {
        local($x1,$y1,$x2,$y2,$n,$r,$i) = @_;
        if($n == 0){ &line(@_[0..3]); }
        else {
                for($i=0;$i<$#lines;$i+=4){
                        &line(&transform(@_[0..3], @lines[$i,$i+1],0),
                              &transform(@_[0..3], @lines[$i+2,$i+3],0));
                }
                for($i=0;$i<$#shapes;$i+=4){
                        &shape(&transform(@_[0..3], @shapes[$i,$i+1],$r),
                               &transform(@_[0..3], @shapes[$i+2,$i+3],$r),
                              $n-1,$r);
                }
                for($i=0;$i<$#rshapes;$i+=4){
                        &shape(&transform(@_[0..3], @rshapes[$i,$i+1],$r),
                               &transform(@_[0..3], @rshapes[$i+2,$i+3],$r),
                              $n-1,!$r);
                }
        }
}
sub line { print "\\drawline($_[0],$_[1])($_[2],$_[3])\n"; }
sub prologue { print "\\begin{picture}(200,200)(0,0)\\thinlines\n"; }
sub epilogue { print "\\end{picture}\n"; }
#
@frame = (0, 0, 200, 200);
@base = (100, 10, 100, 190);
$nest = 5;
while(<>){
        chop;
        if(s/^N\S*\s+//i){ $nest = $_; }
        elsif(s/^B\S*\s+//i){ @base = split; }
        elsif(s/^F\S*\s+//i){ @frame = split; }
        elsif(s/^L\S*\s+//i){ push(@lines,split); }
        elsif(s/^S\S*\s+//i){ push(@shapes,split); }
        elsif(s/^R\S*\s+//i){ push(@rshapes,split); }
}
&prologue;
&shape(@base,$nest,0);
&epilogue;
# program end
@@

このプログラムに対し以下のデータ(\verb+tree+)を与えると
図\ref{tree}のようなフラクタル木が描かれます。

@@ tree
#       フラクタル木のデータ
Base    110 10 110 190
Nest    9
Line    0.0 0.0 0.3 0.0
Shape   0.3 0.0 0.9 0.4
Shape   0.3 0.0 0.9 -0.3
@@

\begin{figure}[h]
\input{tree}
\caption{フラクタル木}
\label{tree}
\end{figure}

また以下のデータ(\verb+fern+)を与えると
図\ref{fern}のような``バーンスレイのシダ''を描くことができます。

@@ fern
#       バーンスレイのシダ
Base    100 10 100 230
Nest    7
Line    0.0 0.0 0.2 0.0
Shape   0.2 0.0 0.96 -0.08
Shape   0.2 0.0 0.4 0.4
RShape  0.2 0.0 0.3 -0.3
@@

\begin{figure}[h]
\input{fern}
\caption{バーンスレイのシダ}
\label{fern}
\end{figure}

これらの処理を自動的に行なうためのMakefileは以下のようになります。

@@ Makefile.sample1
# Makefile
all:
        stangle sample1.sweb
        fractal tree > tree.tex
        fractal fern > fern.tex
        sweave sample1.sweb > sample1.tex
        jlatex sample1
        jlatex sample1
        dvi2ps sample1 > sample1.ps
@@

\end{document}
```

図2: 実験レポートの SWEB テキスト (`sample1.sweb`)

4 SWEB の使用例

ここではデータ及びそれを処理するプログラム、処理結果のグラフ、処理を指示するための Makefile をひとつの SWEB ファイルで扱って実験レポートを作成する例を示す。一般にこのような文書を作成する場合、プログラムやデータを変更するとその度にグラフを書き直す必要があり、レポート内に記述したプログラムやデータとそのグラフの対応を正しくしておくためには細心の注意が必要である。これに対し、プログラムやデータをひとつの SWEB ファイル内で扱い、プログラムの更新や計算、グラフ作成などを Make により自動的に行なわせるようにするとそのような手間が大幅に削減される。ここではフラクタル図形の描画を自動化した例を示す。 SWEB テキストは図 2のようになる。

これを `stangle` に通すことによりこの中に記述された Makefile がプログラムと同じディレクトリに作成されるので、`make` コマンドを起動するとグラフが自動的に作成されて TEX 文書に取り込まれて印刷される。この後 SWEB 文書内のデータを変更したときはもう一度 `make` を起動することにより全ての計算、文書作成処理が自動的に行なわれる。これにより得られた TEX 出力を図 3に示す。



SWEB によるレポート作成

SWEB でレポートを作成する例を示します。描画プログラムとデータがひとつの文書に含まれておりかつ Make によりプログラム / データ / 図 / 文書の作成が自動化されているため、プログラムやデータを修正した場合も再度 Make コマンドを起動するだけで整合性のとれた文書ができあがります。

ここでは与えられたデータに従って各種のフラクタル図形を描画する例を示します。まずフラクタル図形を描画するプログラムを作成します。

図 1: フラクタル木

また以下のデータ (fern) を与えると図 2のような " バーンスレイのシダ " を描くことができます。

#	バーンスレイのシダ
Base	100 10 100 230
Nest	7
Line	0.0 0.0 0.2 0.0
Shape	0.2 0.0 0.96 −0.08
Shape	0.2 0.0 0.4 0.4
RShape	0.2 0.0 0.3 − 0.3

図 2: バーンスレイのシダ

これらの処理を自動的に行なうための Makefile は以下のようになります。

```
# Makefile
all:
        stangle sample1.sweb
        fractal tree > tree.tex
        fractal fern > fern.tex
        sweave sample1.sweb > sample1.tex
        jlatex sample1
        jlatex sample1
        dvi2ps sample1 > sample1.ps
```

このプログラムに対し以下のデータ (tree) を与えると図 1のようなフラクタル木が描かれます。

#	フラクタル木のデータ
Base	110 10 110 190
Nest	9
Line	0.0 0.0 0.3 0.0
Shape	0.3 0.0 0.9 0.4
Shape	0.3 0.0 0.9 − 0.3

1

図 3: 得られた文書出力 (`sample1.ps`)

5 実装と使用実績

sweave, stangle はプログラミング言語 Perl で記述されており、それぞれ約 600 行、約 100 行の小さなプログラムである。全ソースの解説が [6][7] に示されている。SWEB は現在までに大規模プログラムの開発、雑誌記事 [6]、プログラミング言語の解説書 [7]、論文 / テクニカルレポート、実験レポートなど各種の用途に使用されている。本論文も SWEB を使用して作成されている。 [7] では数千行に及ぶプログラムとその解説を全て SWEB で記述することにより SWEB の使用効果が確認されている。

6 既存システムとの併用

6.1 Make との併用

3節で延べたように、 tangle により SWEB 文書を処理する場合、変更の無かった部分に対応するファイルは作成日付が更新されないようになっているので、 Make を使用しても不要な再コンパイル等が起こらないようになっている[†2]。

6.2 RCS との併用

RCS(Revision Control System) に代表されるプログラムのバージョン管理システムは一般にファイル単位でバージョン番号を管理するようになっているため、個々のファイル毎にバージョン番号を付けるのが普通であるが、その場合ファイルの数だけバージョン番号が必要になり、バージョン番号の組み合わせが正しくないと動作しないという問題が発生する。この問題を避けるため、 [2] のような新しいバージョン管理システムでは「バージョングループ」のような概念を使用して複数のバージョンをひとまとめにして扱うことができるようになっているものもある。一方、 SWEB ファイルに対し RCS を使用する場合、個々のプログラムファイルや文書ファイルに対してはバージョン番号が付けられず、ひとつの SWEB ファイル全体に対してのみバージョン番号が付けられるため上述のような問題は発生せず、バージョンの扱いが簡単になる。

6.3 plain2 との併用

plain2[8] はべた書きのテキストを TEX や roff のような文書整形言語の型式に変換するツールである。 plain2 は罫線記号や “-”, “|” のような記号を使って書いた簡易図表を適切な図表作成指令に変換することができるが、SWEB と plain2 を組みあわせて使うことにより、図 4のように、 SWEB 文書中に罫線などを使って書いた図をより美しく整形して出力文書中に含めることができ、 TEX を WYSIWYG 的に使用することができる。

```
\documentstyle[sweb,times,rcsdate]{jarticle}
\long\def\comment#1{}
\title{SWEBとplain2の併用}
\author{増井 俊之}
\rcsdate{$Date: 93/05/20 14:30:06 $}
\begin{document}
\maketitle
以下の図を示すテキストファイルは一度figというファイルに
格納され、plain2により{\TeX}型式に変換されます。その後
\verb+sweave+で得られた{\TeX}ファイルに\verb+\input+で
取り込まれます。
\comment{
@@ sample2fig
 ┌─┬──────┐
 │ 1 │  TeX   │
 ├─┼──────┤
 │ 2 │  roff  │
 └─┴──────┘
@@
}
\input{sample2fig.tex}

これらの一連の処理は以下のMakefileで自動的に実行されます。

@@ Makefile.sample2
all:
        stangle sample2.sweb
        sweave sample2.sweb > sample2.tex
        plain2 -tex sample2fig | plainfix >  sample2fig.tex
        jlatex sample2
        dvi2ps sample2 > sample2.ps
@@
\end{document}
```

図 4: plain2 を併用した SWEB テキスト

7 結論

weave, tangle を使った WEB システムは Knuth の理想を追及したものであり一般のソフトウェア開発において実用的とは言い難かったが、既存のソフトウェアツール環境にあわせて簡単化 / 拡張した SWEB システムは各種のソフトウェアや文書作成に有効利用できる。 SWEB は単純な仕様にもかかわらず WEB の特徴を保っており、Make, RCS, plain2 などの既存のツールと併用してさらに有効な使い方が可能である。今後はマニュアルやヘルプファイルなどの作成にも使用して評価を行なってゆく予定である。

参考文献

[1] Donald E. Knuth. Literate Programming. *The Computer Journal*, Vol. 27, No. 2, pp. 97–111, 1984.

[2] Andreas Lampen. Advanced files to attributed software objects. In *USENIX Conference Proceedings*, pp. 219–229. USENIX' Conference Proceedings, Winter 1991.

[3] Silvio Levy. WEB adapted to C: Another approach. *TUGBoat*, Vol. 8, No. 1, pp. 12–13, 1987. TEX のディストリビューションに付属.

[4] Norman Ramsey. Weaving a language-independent WEB. *Communications of the ACM*, Vol. 32, No. 9, pp. 1051–1055, September 1989.

[5] H. Thimbleby. Experiences of 'Literate Programming' using cweb (a variant of Knuth's WEB). *The Computer Journal*, Vol. 29, No. 3, pp. 201–211, March 1986.

[6] 増井俊之. Perl 書法 – perl で作る WEB. *Super ASCII*, Vol. 3, No. 6, pp. 126–135, June 1992.

[7] 増井俊之. Perl 書法. アスキー出版局, 1993. to appear.

[8] 内田昭宏, 佐野晋. べた書き入力による文書清書の試み. 第 32 回冬のプログラミングシンポジウム予稿集, pp. 109–120. 情報処理学会, January 1991.

[†2] オリジナルの WEB をはじめ CWEB, cweb, Spider などではこのような考慮はされていないため、文書部分だけを変更した場合でも全プログラムの再コンパイルが行なわれてしまう。