構造化文書作成支援システムとしてのViVi, vineについて

津田伸秀[*1]

研究者，エンジニアは日々構造化文書を作成しており，それを効率化し知的生産性を向上する必要がある．しかし従来のシステムはファイル普遍性，広義の可読性に劣り，編集の操作性が優れないなどの問題があった．本稿で紹介するシステムは，エディタ ViVi，レイアウトソフト vine の2つのプログラムから構成され，プレインテキスト，行頭記号記法，アウトライン表示，エディタからの機能拡張アプローチ，GUI によるスタイルセットの設定機能などの特徴をもっている．多くのユーザが本システムを実務に使用しており（本稿も本システムで作成している），知的生産性を向上させる一定の効果をあげている．本システムの開発には関係性開発モデルを採用し，ユーザとの直接的なコミュニケーションによりオープンで効率的な開発を実現している．

はじめに

みなさんはこの文章のような原稿を作成する場合，どのようなツールを使用されているでしょうか？一般的には MS Word などのいわゆるワープロソフトを使う方が多く，研究者の方に限って言えば TeX[1]を使用される方が多いと思います．しかし，ワープロソフトは素人でも簡単に使用できる反面，重遅で使い勝手が悪く，知的生産的の高いツールとは言い難いと考えます．TeX は高度な機能を持っており，美しい印刷物が得られるすばらしいものですが，素人には敷居が高く習得も大変で，使い勝手が悪い面もあります．

そこで筆者はそのような問題を解決するために，ワープロの簡便さと TeX の柔軟性と厳密性を持ち，かつ処理能力の低いマシンでもストレスなく使えるパフォーマンスを持った構造化文書作成支援システムを研究開発中です．システムは ViVi と vine の2つのプログラムで構成され，Windows 95/98/NT で実際に使用可能です．本稿ではこれらの簡単な紹介およびその開発モデルの説明を行います．

なお，本稿は本システムを用いて作成しています．

構造化文書作成支援システムの要件

長文作成においては文書構造を明確にすることが必須です．全体をいくつかの部分にまとめ構造を整理することで，読者，文書作成者の文書理解を容易にする必要があるからです．その結果，整理された文章は階層構造を持つことになります．このように文書構造を明確に意識して作成され，文書構造を明示した文書を**構造化文書**と呼びます．たとえば，この文章や学術論文，長文のレポート，報告書，小説などは構造化文書です．

構造化文書の作成を効率的に行うことを目的とした構造化文書作成支援システムは，(1) ファイル普遍性がある，(2) 広義の可読性に優れる，(3) 編集の操作性に優れる，という条件を満たす必要があると考えます．

● ファイル普遍性

一般に使用されているワープロソフトは複数の種類，バージョンが存在します．文書ファイルはそれを作成したソフトの特定のバージョンでないと編集できない場合が多く困ってしまいます．これは文書ファイルがバイナリ形式で固有の構造を持っているからです．

プレインテキストであれば，どのようなＯＳでも読み書きできる場合が多く，普遍性の問題を回避できる場合がほとんどです．プレインテキストであってもなんらかの書式で構造を記述する以上，書式が異なるソフト間で互換性があるわけではありませんが，バイナリ形式に比べ，フィルタやエディタで書式を変換することが容易です．また，プレインテキストを編集するエディタは多種多様のものが出回っており，ユーザの好みのものを使用できるという利点もあります．

● 広義の可読性

構造化文書は以下に示す4つの要件からなる広義の可読性を満たす必要があると考えます．

構造化文書をプレインテキストで表現する方法としては，SGML[2], TeX 等があります．これらは文書構造を指定するためにタグをソース文書に挿入しますが，文章を推敲する時にそれが思考を中断するのでソース文書の可読性を低下させます．文書作成者は文書構成，表現を考えながらソース文書を編集するため，ソース文書の可読性（これを狭義の可読性と呼ぶ）は構造化文書作成効率に大きく影響を及ぼす要件です．

次に，文書を記述している時は局所的な思考に陥りがちで，全体の構成に対する位置づけを忘れがちになります．よって，文書構造のアウトラインを容易に表示できるかどうかは大変重要です．

さらに，人間は文書全体の構成を把握することも，局所的な部分を把握することも可能ですが，それを同時に行うことはできないので，それらの視点を切り替えること（視線の往復[3]）を容易にする必要があります．

最後に，アウトライン表示が可能であっても画面の情報量は限られているので，推敲は印刷結果で行った方が効率が上がります．奇麗な印刷が可能かどうかも重要です．

ここに示した，狭義の可読性，アウトライン，視線の往復，奇麗な印刷は他人が記述した文書の理解を容易にするのはもちろん，文書作成者の思考支援機能としても大変重要な要件です．広義の可読性は文書の楽読性とも言えるもので，文書理解のコストの低さを示すものです．

● 編集の操作性

構造化文書を作成する場合においても，文書構造の編集を行うよりも，本文の編集に多くの時間を費やします．したがって，（あたりまえのことですが）編集の操作性が優れているかどうか

*1 ソフトウェア作家，佐賀大学理工学部 客員助教授，
mailto:ntsuda@beam.ne.jp

が構造化文書作成功率に大きく影響します．現状のアウトラインプロセッサは編集機能が貧弱なものが多く，ワープロは規模が肥大しパフォーマンスが劣るという問題もあります．

ViVi方式

上記のような問題を解決するために筆者らはViVi方式による構造化文書作成支援システムを提案[4]，一般に公開[5]してきました．本方式は，(1) プレインテキスト，(2) 行頭記号記法，(3) 行頭記号入力支援，(4) アウトライン機能，(5) スタイルファイルを用いたレイアウト&印刷機能から構成されます．

● プレインテキスト

本システムの文書作成の流れはTeXと同様にViViなどのエディタでソース文書を特定の記法で記述し，それをvineでレイアウト&印刷するというものです．よってソース文書はプレインテキストになります．これによりファイル普遍性が向上し，他のＯＳとの文書のやり取りも楽になります．

● 行頭記号記法

段落の先頭に“■”や“○”等の記号を記述することで段落種別を明示する記法を**行頭記号記法**と呼びます．日本の学術雑誌，書籍でも普通に用いられている記法です．“<H1>”や“\section”のような記法に比べ記号1文字の方が認識が容易で意味をシンボライズしているのでソース文書の可読性に優れ，一般人がなじみやすい記法であると思います．

“第1章”，“1.2”のような連番も広義の行頭記号とみなします．連番接頭辞，接尾辞や，どの記号を行頭記号として認識するかはViViのプロパティで容易に設定可能です．

一般に流通しているソフトのreadme.txt等のプレインテキストの構造化文書は，この行頭記号記法に則っている場合が多く，ViViで見ると構造の把握が楽になります．

図1は本稿の実際のソース文書です．前ページのレイアウト結果と見比べてください．最初の段落はタイトル，次の段落は著者(脚注については後述します)，3番目はアブストラクトとなっています．“■”は大見出しを表す行頭記号です．図2にTeXで記述した場合のソースを示しますが，比べてみると行頭記号記法で記述したソース文書がいかに可読性に優れているかを実感していただけると思います．

● 行頭記号入力支援

ViViでは単手数のキー操作(Alt+矢印キー)で行頭記号/連番の入力/修正を可能にしました．“1.2”，“1.2.3”のような枝番も単手数で入力可能です．頻繁に行う操作を単手数で行うことを可能にすることは習熟者の生産性を上げるのに大変効果があると考えています．

● アウトライン機能

図3に示すように，ViViはソース文書を行頭記号，連番に注目してパースすることで文書構造を抽出し，画面左のペインにツリー表示することができます．マウスおよびキー操作によりノードの展開，折り畳み，ノードに対応する本文位置へのジャンプ，アウトラインでの編集，リナンバ，アウトライン表示のＯＮ・ＯＦＦが可能です．

アウトライン機能により文章構造の把握が楽になるとともに，視線の往復もキー操作のみで可能になりました．

```
構造化文書作成支援システムとしてのViVi, vineについて

津田伸秀≪ソフトウェア作家，佐賀大学理工学部 客員助教授，¥nmailto:ntsuda@beam.ne.jp≫

　研究者，エンジニアは日々構造化文書を作成しており，それを効率化し知的生産性を向上する必要がある．しかし.....

■ はじめに

　みなさんはこの文章のような原稿を作成する場合，どのようなツールを使用されているでしょうか？一般的には.....
```

図1 本稿のソース文書

```
¥documentclass[12pt,twocolumn]{article}
¥title{構造化文書作成支援システムとしてのViVi, vineについて}
¥author{津田伸秀}
¥begin{document}
¥maketitle
¥begin{abstract}
　研究者，エンジニアは日々構造化文書を作成しており，それを効率化し知的生産性を向上する必要がある．しかし.....
¥end{abstract}
¥section{はじめに}
　みなさんはこの文章のような原稿を作成する場合，どのようなツールを使用されているでしょうか？一般的には.....
```

図2 TeXによる本稿のソース文書

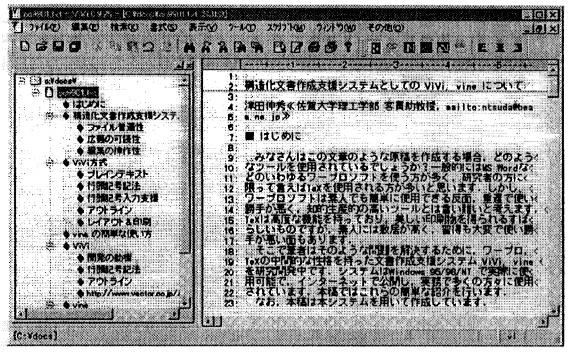

図3 ViViの画面

図4 vineの画面

● レイアウト&印刷

レイアウトと印刷はエディタとは独立したプログラムvine(図4参照)で行います．左のペインでアウトラインを表示し，右のペインにレイアウトされた文章を表示します．

vineの特徴は，行頭記号記法により段落の種別を指定し，それぞれの印字スタイルをＧＵＩにより簡単に設定できること

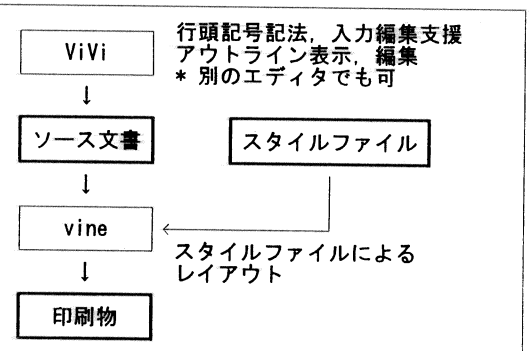

図5 ViVi, vine による構造化文書作成

です(詳細は次章参照)．本文，タイトル，大見出しなどの段落種別は，それぞれにフォント名，フォントサイズ，アライメントなどの情報を持ちます．これをスタイルと呼びます．ソース文書では行頭記号記法により段落種別を指定し，スタイルファイルにある各段落種別に対応する印刷スタイル情報が参照されてレイアウトが行われます．

ユーザが定義したスタイルセットに則って記述されたソース文書を vine にドロップすると，ページ設定，組み版ルール，各種スタイルの設定にしたがってレイアウトが行われ，画面に表示されます(図5参照)．レイアウトを画面 and/or 印刷結果で確認し，修正する個所があればツールバーのボタンを押してエディタを呼び出し，ソース文書の修正を行います．その後 vine にフォーカスを戻すと，ソース文書を再読み込みするかどうかを尋ねてくるので【はい】を選べば，再レイアウトを行います．

実際の文書は文書構造単位のスタイルだけでは不十分で，強調文字などを入れる機能は必須です．vine ではそれをインラインスタイルと呼び，本文に埋め込む形式のインラインマークアップで指定します．

インラインマークアップとは SGML の様にマークアップ開始文字列と終了文字列でスタイル種別を指定する方式です．たとえば図1では“≪”，“≫”を脚注のマークアップに使用しています*2．インラインマークアップに使用する文字列はスタイル設定で自由に指定できます(たとえば<note>,</note>)．

段落スタイルはユーザが登録修正し，スタイルファイルとして保存することでライブラリ化することができます．

● まとめ

表1に従来の構造化文書作成支援システムの問題点と ViVi 方式の解決方法をまとめておきます．

表1 問題点と解決方法

問題点	解決方法
ファイル普遍性	プレインテキストの使用
広義の可読性	行頭記号記法 アウトライン表示 奇麗なレイアウト，印刷結果
編集の操作性	エディタからのアプローチ

*2 このような表記はソース文書の可読性を損なうのではと心配されるかもしれませんが，ViViであれば，脚注部分の表示色を本文と異なるものに設定することで，可読性を損なわないようにできます．

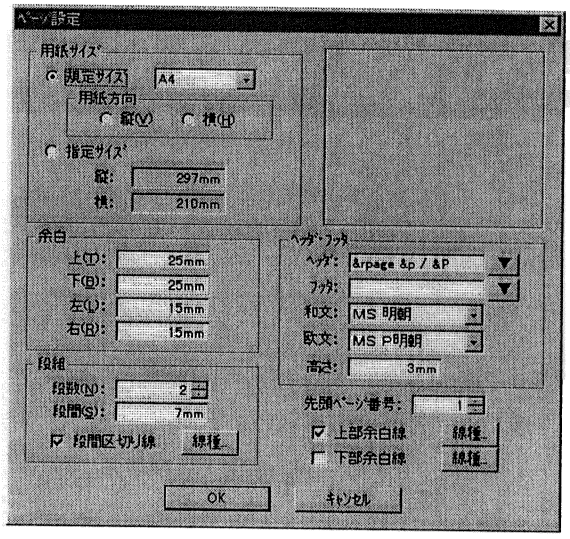

図6 ページ設定

vineの設定について

本稿で ViVi, vine の詳細をすべて説明するには紙面も筆者の時間も充分ではありませんので，本章では vine のページ設定，組み版ルール，スタイル設定についてのみ少し詳しく説明します．その他の点については公開しているファイルに同梱されているドキュメントを参照してください．

● ページ設定

ページ設定ダイアログを図6に示します．このダイアログは「設定」-「ページ設定」メニュー，または画面の余白部分をダブルクリックすることで呼び出すことができます．

ページサイズ，用紙方向，上下左右余白サイズ，段組指定，ヘッダフッタ文字列，ページ番号初期値等の設定が可能です．ヘッタフッタにはページ番号，ファイル名，印刷日付時間などを左右中央揃え指定付きで設定できます．

図6では長さの単位にミリが使われていますが，この他にポイント(point)，インチ(inch)，級(Q)が使用できます．

ダイアログ右上にはページのイメージが表示される予定ですが，現段階ではまだ実装していません．

● 組み版ルール

組み版ルールの設定ダイアログを図7に示します．このダイアログは「設定」-「組み版ルール」メニューで呼び出すことができます．

禁則処理のＯＮ・ＯＦＦ，ぶら下げ可能文字，行頭，行末禁則文字，両端揃え，英字ワードラップ，和欧間空白の指定ができます．

オプションとしては指定できませんが，vine のレイアウトエンジンは連続役物2分処理も行っています．ここまでちゃんと日本語組み版処理しているものはワープロソフトにもそう無いはずです．これらの機能により奇麗なレイアウト結果を得ることができるようになりました．

● スタイル設定

「設定」-「スタイルプロパティ」を実行すると，図8のプロパティシートが開きます．プロパティシートとは図の様に上部

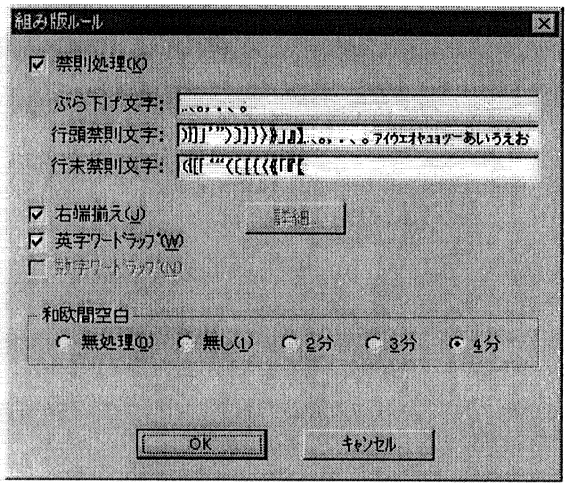

図7 組み版ルール

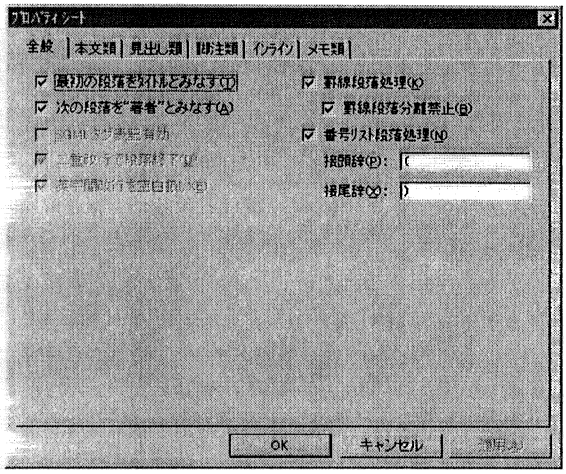

図8 スタイル設定

にタブと呼ばれる見出しのようなものがあり，その部分をクリックすることによりページを切り替えることが可能なダイアログのことです．

デフォルトのページではソース文書の最初の段落をタイトルとみなすかどうかの設定などができます．本分類，見出し類などのそれぞれのスタイルは上部のタブで切り替えられるページで設定を変更できます．

図9は本文スタイルの設定ページです．マークアップ書式は該当スタイルを示す行頭記号を指定するものです．たとえば一覧リストであれば“・”を指定するようにします．記号だけでなく“<li>”などの文字列を指定することも可能なので，部分的にSGMLなどと互換性のあるマークアップ書式を設定することができます．

マークアップ書式で指定された段落は，ここで指定されたスタイルでレイアウトされます．スタイルには左右余白，水平アライメント，タブ位置，フォント種別などがあります．フォント種別は和文，欧文フォントフェースを別々に指定可能で，ボールド，イタリック，アンダーラインの指定も可能です．

図10は見出しスタイル設定ページです．見出し類のスタイルは本文類にはない設定項目がいくつかあります．印字書式で“1.1”などと指定することで章節番号を自動生成することが

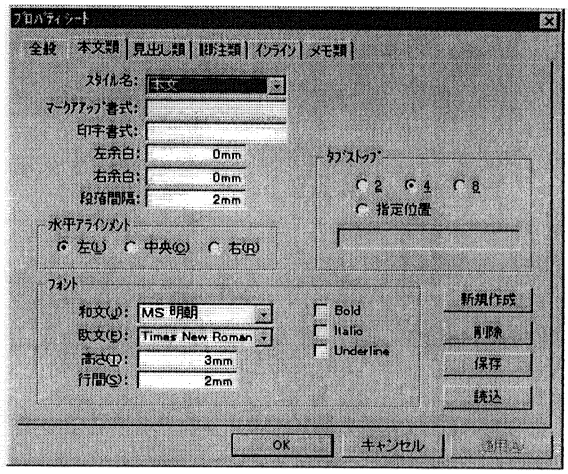

図9 本文スタイル設定

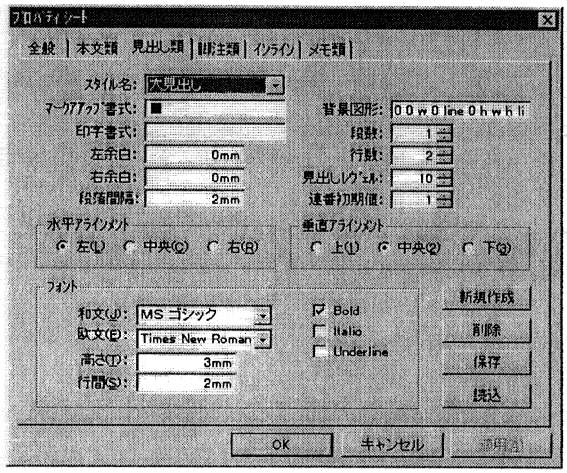

図10 見出しスタイル設定

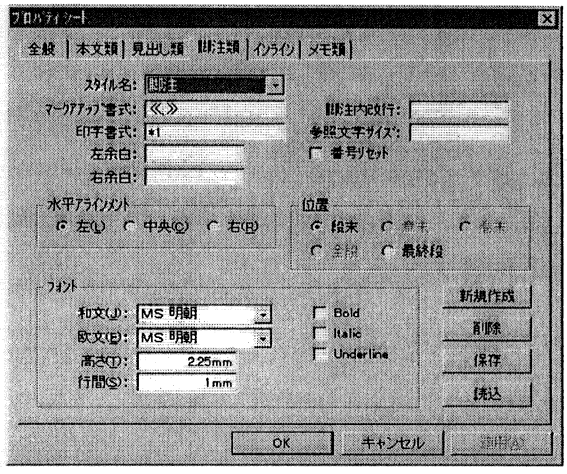

図11 脚注スタイル設定

できます．背景図形で見出しの部分の背景に表示する図形を定義できます．本稿では上下に水平線のある図形を使用しています．見出しは1行で本文n行分のスペースを確保することができます(本稿は2行分です)．そして，その中での垂直アライメントの指定ができます．段数も指定できますが，現在はタイトル，著者，序文以外は1段のみをサポートしています．

図11は脚注スタイル設定ページです．脚注はエディタへの

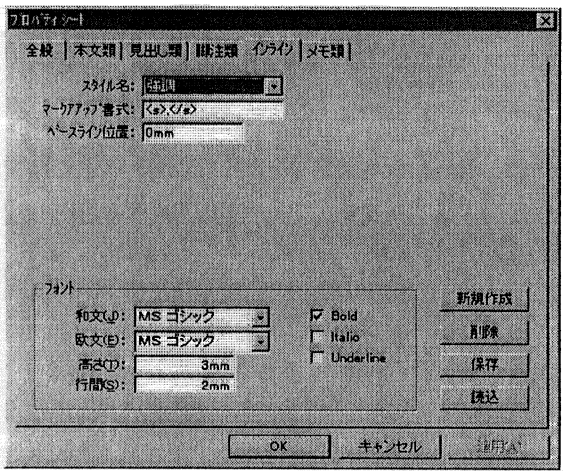

図12 インラインスタイル設定

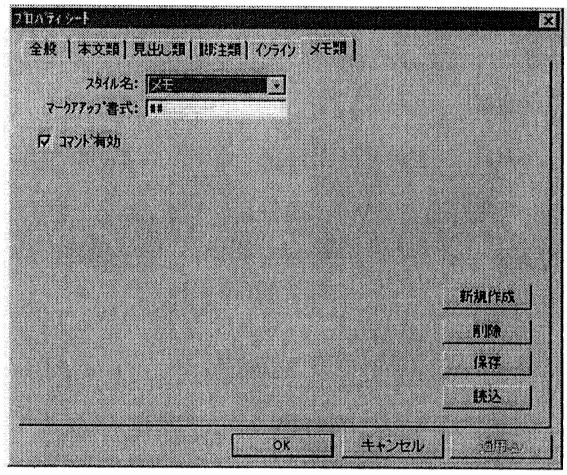

図13 メモスタイル設定

要望としてはかなり大きなもので，これをサポートしていることはvineの大きなセールスポイントです．マークアップ書式で脚注の書式を指定し，脚注の表示位置などを設定できます．印字形式では，本稿の様に「記号＋数字」だけでなく，指定記号を脚注番号数印字するよう指定することもできます．

図12はインラインスタイル設定ページです．強調文字や上付き下付き文字などはインラインスタイルで指定します．インラインスタイルは本文に埋め込まれるので，通常のマークアップ言語のように，開始文字列と終了文字列で指定します．区切りにはカンマを使用します．

上付き文字を定義する場合は，ベースライン位置を指定し，本文文字に対し上方に配置されるようにします．ベースラインを下げることで“$\mathrm{T_EX}$”の様なレイアウトも可能です．

図13はメモスタイル設定ページです．メモは印刷されない段落です．一部をコメントアウトしたり，履歴を記述することができます．

『コマンド有効』オプションをＯＮにしておけば，表2に示す特殊コマンドが使用可能です．

● まとめ

vineは様々なスタイルをＧＵＩで自由かつ簡単に設定することができます．作成したスタイルはテキストファイルに保存でき，ライブラリ化したり，ユーザ間で共有することができます．使用するスタイルが決っていれば，後は本文をエディタで作成するだけなので，構造化文書を効率よく作成することが可能になります．

表2 コマンド一覧

機能	コマンド
改段	newColumn
改ページ	newPage
行間挿入	lineSpace
行の重ね打ち	noLineFeed
bmp 表示	img=″name.bmp″ width=90% caption=″キャプション″ pos={here|top|bottom}
表組み	table
コラム	column

関係性開発モデル

本方式はViVi, vineに実装し，Windows上で実動するシステムとしてインターネット[5]，書籍，雑誌などにより広く一般に公開しています．学生が学術利用する場合はユーザ認証を受ければ無料で使用できるのですが，1999年1月末時点での学生認証者数は約1500人です．ユーザアンケートによれば学生の割合は7～10%なので，全ユーザ数は約2万人と推定しています．

開発言語はC++で，クラスライブラリとしてMFCを使用しています．システム全体のソースコードは約10万行です．筆者は20年以上もソフトウェアの研究開発を行ってきましたが，一つのプロジェクトで自分一人が担当した部分としてはもっとも規模の大きいものになってしまいました．

このプロジェクトの名前は「売れない作家プロジェクト」といい，1996年4月末にWindowsプログラミングの勉強のためにエディタを作り始めたのが起源です．その後紆余曲折があり今だに完成版とは宣言していないにもかかわらずViViは高い評価をいただき，日本中のとても多くの方々に使用していただけるほどに育ってきました．これは試用版をインターネットにより公開し，それを試用するユーザの方々と直接的なコミュニケーションをとってきたのが功を奏していると考えています．このような形態は関係性マーケティングと呼ばれるようですが，開発モデルとしては経済活動を伴わない場合もあるので筆者は関係性開発モデル(Relationship Development Model)と呼んでいます．

関係性開発モデルでの開発は，試用版の作成と配布，ユーザとの直接的なコミュニケーション(Mailing List, BBS, OFF会)による要望の吸い上げ，ユーザの実環境でのテスト，継続的な機能拡張，品質向上という流れになります(図14参照)．このモデルによりユーザのニーズを的確にそして効率よく理解することが可能になります．ソフトウェア開発における品質保証(Quality Assurance)工程では，テストのために様々な環

境を用意できない場合が多いのですが，実際のユーザに実務でテストしていただけるので，テストの精度が高いとも言えます．さらに，開発状況がオープンなので，ユーザ自身が開発状況を判断できるという利点もあります．

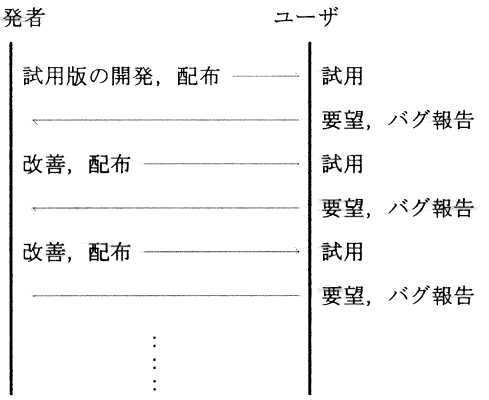

図14 関係性開発モデル

ViViの場合，1996年12月からメーリングリストを立ち上げ，関係性開発モデルによる開発を行ってきました．当初，メンバ数は100人程度でしたが，2年たった現在では550人ものメンバがいます．メールの流量は作者がどの程度対応するかで大きく変わってきます．1日に20～30通のメールが配信される場合もありますが，1999年1月末までのメール総数は約8000通なので，平均すると1日約10通になります．

多人数に向かって発言することは敷居が高い面もあるので，ほとんど発言を行わずメールを読むだけの方も少なくありません．そこで，そのような方の意見，要望を吸い上げるためにユーザアンケートを年に2～3回実施しています．

ViVi, vineではこの関係性開発モデルにより，ユーザの要求を効率的に把握できていると考えています．しかし関係性開発モデルにも問題点が無いわけではありません．多くの要望がよせられるが，どの要望を採用するかの判断は非常に難しい場合があります．相矛盾するものや，要望がよく理解できないもの，実現するための工数があまりにも大きいものや，作者の趣味でなく気乗りしないものもあります．それらへの対応は慎重に行う必要があります．ユーザの要望は限りがないことも問題です．すべての要求に応えるとバイナリが肥大化し，パフォーマンスが低下します．ユーザは自分の欲しい機能は実装して欲しいが，いらないものは実装してほしくないものです．また，すべての要望を受け入れているとソフトの方向性が曖昧になり，特定のユーザ層のニーズに即した物という定義に反するようになります．筆者はこれを八方美人問題と呼んでいます．

これらの問題をうまくさばくには，作者の判断能力，決断力が要求されます．しかし，関係性開発モデルは，開発者ユーザ双方に大きなメリットがあり，不特定多数向けソフトウェア開発では今後ますます採用されていくものと考えています．

おわりに

筆者らは一般文書，プログラムソース等の構造化文書の効率的な作成を助け，知的生産性を向上させるためのシステム；ViVi, vineを研究開発しています．関係性開発モデルを採用することでオープンで効率的な開発を行っています．まだまだ開発段階であり完成版と呼べるものをいつリリースできるのか未だ良く分からない状況ですが，多くのユーザが実務に使用し，たくさんの意見，要望，問題レポートを送ってくださっています．それによりシステムは日々進化を続けています．もしWindowsを利用可能でしたら，ぜひ一度ご試用してみてください．実際に試した上での，ご意見，ご要望，バグレポートをお待ちしております．

本システムが多くのユーザに受け入れられるようになってきたのはＭＬ等でお世話になっているユーザ各位に負うところが大きいと考えています．彼(女)らには大変感謝しており，今後もこの良い関係を保っていけるよう努力していきたいと考えています．

【参考文献】

[1] Donald E. Knuth: http://Sunburn.Stanford.EDU/~Knuth/
[2] http://www/isgmlug.org
[3] Hans Greorg Gadamer: *Wahrheit und Methode; Grundzuge einer* philosophichen Hermeneutik, Tubingen, Mohr, 1960
[4] 津田，若松：*[6Q-08] 行頭記号を用いた構造化文書作成支援システム*について，情報処理学会第57回全国大会，1998
[5] 津田：http://www.vector.co.jp/authors/VA007799/