保守を考慮したソフトウェア開発環境 MOSDENV

おさだかずひさ　たなかひでひこ　もとおかとおる
長田 和久,　田中 英彦,　元岡 達
(東京大学工学部)

1．はじめに

増大するソフトウェアコストを低減するためには，開発コスト，ドキュメント，改版のしやすさ，ユーザーインターフェースなどさまざまな問題を解決しなければならない。なかでも「保守」のコストは，相当な部分を占めている（といわれている）。すなわち，予防保守をしつつ効果的なソフトウェア開発を支援することが必要である。

MOSDENV (Maitenance Oriented Software Development ENVironment) は，2節で述べるように，改版が数多くなされるような（あるいは，開発初期にプロトタイプが種々作られるような）場合を想定している。すなわち，初期投資が多少割高でもトータルでひきあうことをねらう。

トップダウン設計・モジュール分割による（木構造を基本とする）ドキュメント方式と，マルチウィンドウエディタを対応させること（図1）を主眼とした実験システムを実装したので紹介する。あわせて，ソフトウェア開発過程の螺旋型モデル表現の紹介も行う。

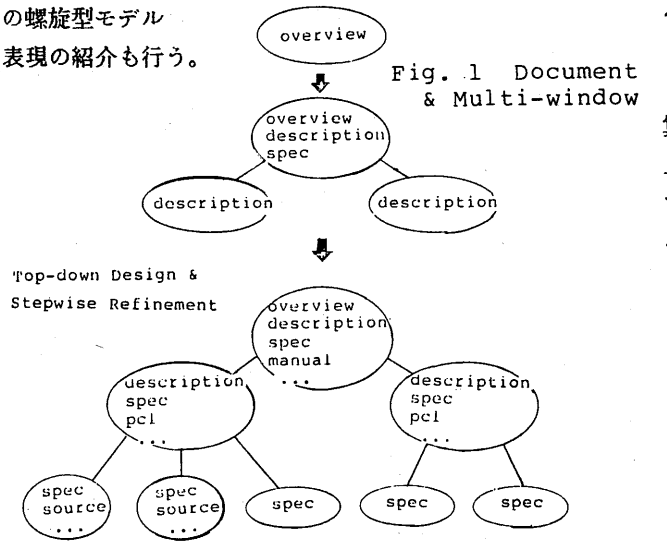
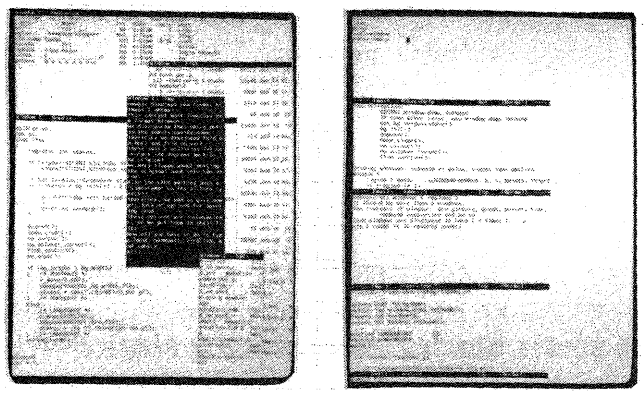


Fig. 1 Document & Multi-window



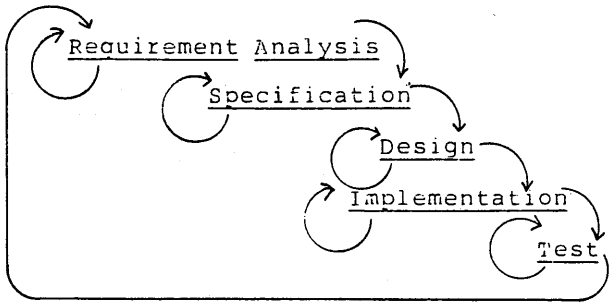


Fig. 2 Typical Software Life Cycle

2．ソフトウェア開発過程の螺旋型モデル

近年，ソフトウェア開発方法論のひとつとしてラピッドプロトタイピングがソフトウェアライフサイクルモデル（図2）に対してとりあげられている。〔3，6〕 現在のところ「ライフサイクル」は開発管理のためにあると同時に現実に存在するものであり，一方「ラピッドプロトタイピング」は，開発方法論のひとつ（ツールのひとつ）であるという見方が一般的である。

ラピッドプロトタイピングの正確な定義はまだないが，「実働可能なモノをすばやく安上がりに（開発の初期に予算の数割程度で）作成し，ユーザの要求を明確にする手法」といえよう。さらに，できたプロトタイプを「捨ててしまう」場合と「繰り返し改良していって最終的に製品とする」場合とがあろう。

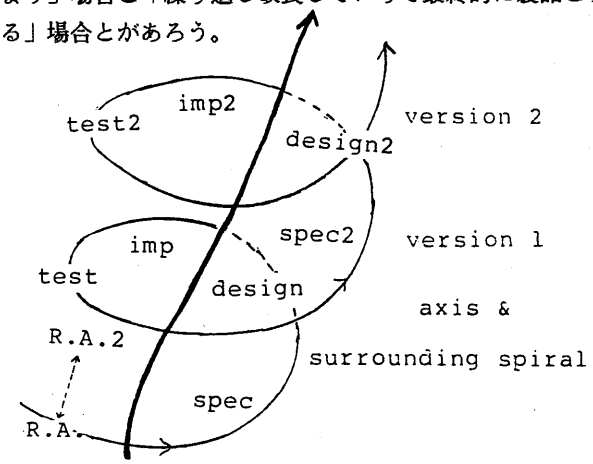


Fig. 3 Representation of Spiral Model

ラピッドプロトタイピングとソフトウェアライフサイクルモデルは相反するものであるという議論〔4，5〕もあるが，むしろラピッドプロトタイピングをライフサイクルモデルにとりこむことが可能ではなかろうか。すなわち，evolutionary, incremental or iterative development を表現する以下のような「ソフトウェア開発過程の螺旋型モデル」を考えてみる。

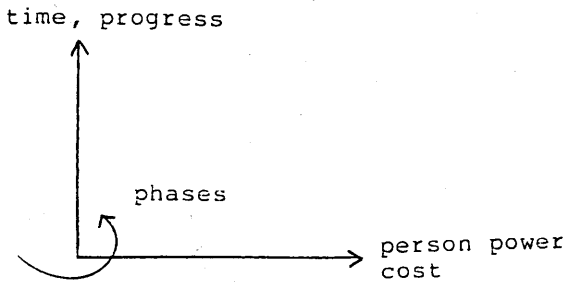


Fig. 4 Parameters of Spiral Model

螺旋型モデルは，「軸」とそれをとりまく「螺旋」によってソフトウェア開発過程を視覚的に表現する方法である。（図3，4）軸は分岐・合流・その太さにより，設計思想と製品の状態とを表す。（図5）螺旋は，軸からの距離でコスト，回転角で各フェイズ（の進度）〔一回りでひとつの版またはプロトタイプができるとする〕，軸に沿った距離は経過時間（所要時間）をそれぞれ表す。3次元的に示すことで各フェイズの各版間の繋がりも表すことができる。

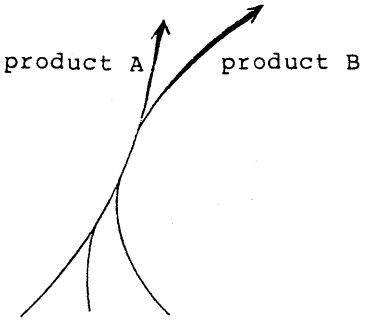


Fig. 5 junction & confluence

螺旋型モデル（表現）を使うことによって開発過程を視覚的にとらえることができる。たとえば（図6）同じ「人・月」のコストで作成した場合でも１００人×１月と１０人×１０月の違いを表現できるわけである。(a)は多人数短期間，(b)は少人数長期間の場合を示している。

さらに，ラピッドプロトタイピングの場合は，開発初期にプロトタイプが繰り返し作られるため図7のように表現できる。このときプロトタイプと第1版とでは，螺旋の各フェイズのようすがちがう。（e.g. 一周のコストは低いが要求分析の比重大）

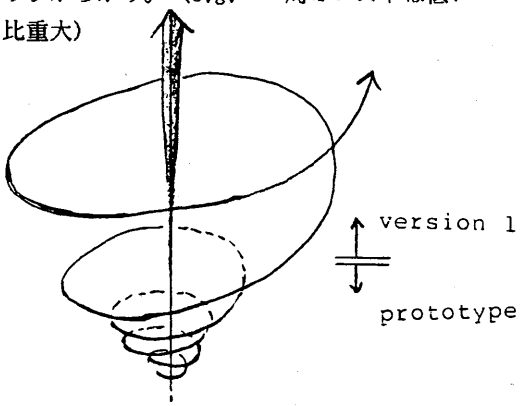


Fig. 7 Spiral Model of Rapid Prototyping

３．MOSDENVの構成

３．１ システムの構成

図8に示すように，**MOSDENV** は，「ソフトウェア開発方法論」，「保守用データベース」，「ユーザーフレンドリな環境」という3つの側面から“環境”をとらえて，それらを基本方針としている。さらに，4節で述べるようにツールは，

・ドキュメンテーション方式〔1〕
・プロンプティングシステム
・改版支援ツール

のようなものがある。

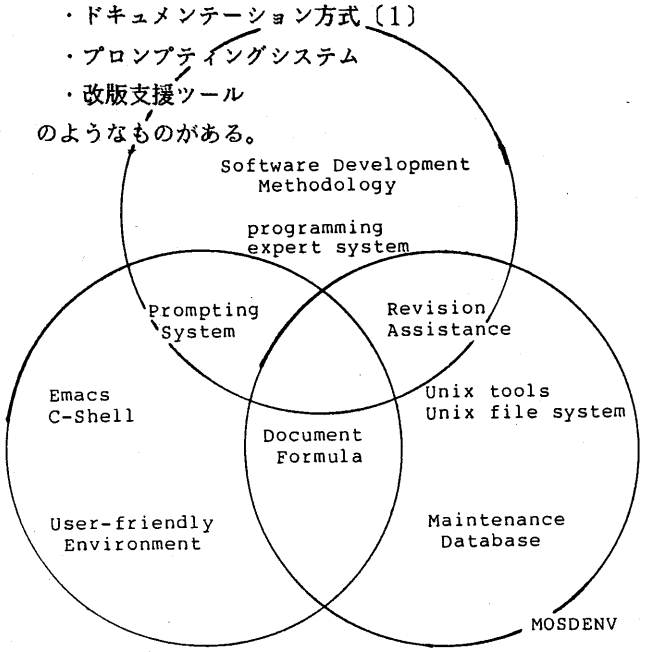


Fig. 8 the basic plan of MOSDENV

今回の実験システムでは，Unix TM とマルチウィンドウエディタＥｍａｃｓ（図8の〔括弧〕内）をもとに，ハードウェア構成を図9のようにした。Mainframes間はネットワークコマンド cvos，cvos2，cu，cx 等でつなぐことができる。保守支援されるソフトウェア記述言語は“Ｃ”，“C-Shell Script”である。

Unix は AT &T (Bell研) の登録商標。

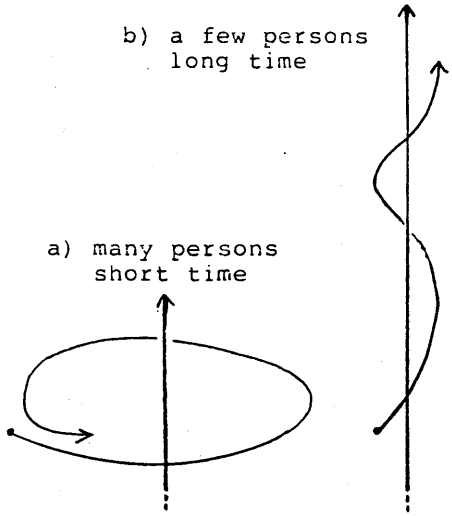


Fig. 6 Examples of Spiral Model

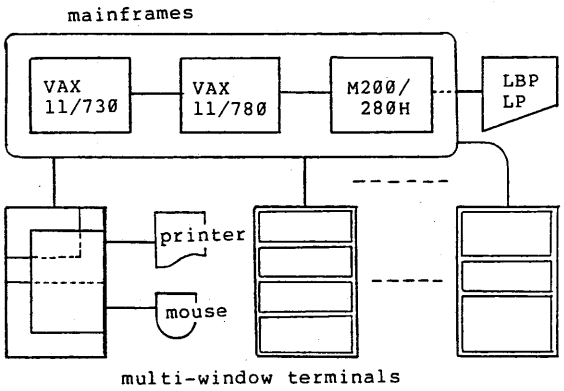


Fig. 9 Machines configuration of MOSDENV

3．2 MOSDENVの特長

(1) 初期開発時には，プロンプティングシステムと対話的に操作することにより，階層設計・段階的詳細化 topdown & stepwise refinement を支援する。

(2) トップダウン設計・モジュール分割に自然に対応したドキュメンテーション方式によりソフトウェアが構成される。

(3) 各々のモジュールのドキュメント群の関係，またモジュール間に module dependency，time dependency，を決めることにより，改版支援を行う。

(4) 既存のツール群の積極的活用。
Unix，Emacs，etc.

(5) マルチウィンドウとドキュメンテーションの自然な対応によるユーザーインターフェースの向上。

3．3 動作の具体例

ある種のＣＲＴでは倍幅文字や倍高文字を出力することができる。例題としてこのプログラムを作成する場合をかんがえる。

example

MDISP - modified displayer

① overview を記述する（数行）たとえば，
“support DOUBLE Width／Heigt Line command. etc.”

②ウィンドウが２つになり description を他のウィンドウに記述する。〔やや詳細に〕「標準入力から一行とり所要のコードを付与して標準出力に出す」

③ spec を記述，各種の escape sequence などにふれる。
〔description と異なり形式的に書く―擬似コード，Shell Script〕
DHDW - double height double width
DHSW, SHDW, SHSW, などをフラグで区別できる。

④ MDISPの使用法を同時に考えることにし，manualのウィンドウを開いてフラグの意味を決める。
w - double width, h - double height
Usage : mdisp〔wh〕〔files ... 〕

※ overview，description，spec，manualの４つのウィンドウが同時に見えかつマウスで自由に移動・編集できる。

⑤各々のドキュメントを煮詰め，spec が定まるとプロトタイプを作る。これには言語Ｃで書く前に “AWK”（Unix のツールのひとつであるパターンスキャナ）を使用した。

※画面には，specとsource（awk）とawk のマニュアルのウィンドウがある。

⑥ source および awkのサブファイルをいくつか作成した後テストする。この典型的なテストパターンを pcl（program check list）に記述する。

⑦作成途中で記録しておきたいことは memo に随時書く。

⑧プロトタイプが完成した時点で各ドキュメントを固定し，以後の変更は，いったんすべて overview まで戻ることになる。

※以上の手順を支援するものがプロンプティングシステム（4.3 節）である。

⑨ awkによるプロトタイプではやや実行時間がかかるため言語Ｃによる第２版を作成することにする。
（mdisp.1 → mdisp.2）

⑩第１版の overview，descriptionが画面に出る。これらに必要事項を追加して第２版のoverview，description とする。〔ほとんど変更なし〕

⑪順次 spec，manual 等を変更する。たとえば，
フラグ w → -w （Unix流）

⑫ source をＣで記述する。

※ spec，source，Help，Manual などが画面上にはある。

⑬ pcl，objectを作成しテストする。

⑭ＣＲＴによって escape sequenceが異なるため数種のＣＲＴでも動作可能なように改版することにする。

※いきなり source を編集しなおすのではなく，最新版の overview から順次（複数ウィンドウなのでほぼ同時に）ドキュメントを再読してゆく。

⑮ overview は変更なし，description 以下を変更する。第３版の完成である。

※このような改版を支援するものが“改版支援ツール”（4.4 節）である。

※各種ドキュメント群は，直接個々のものを見ることも可能だが適当な基準で再構成することもできる。（4.2 節）

拡大 ＡＷＫ ａｗｋ
標準 AWK awk

Fig. 10 An Example

この例の場合は，簡単のためひとつのモジュールだけのドキュメント構成とその作成手順を示した。複数モジュール（ひとつのモジュールはひとつのコンパイル単位）の場合は，モジュール間の関係を示すファイルである“Table”，“Where”，“interface”などが重要な役割を持つ。

4．MOSDENV の実験システムと評価

4．1　基本ツール

実験システムの基本的環境・ツールとして

・Unix TM　とそのファィルシステム
・マルチウィンドウエディタＥｍａｃｓ
・ビットマップディスプレイ
・ポインティングデバイス—マウス
・ウィンドウシステム

を使用した。それぞれ具体的には—

Unix 4.1 BSD （ウィンドウマネージャーとネットワーク機能をサポートするという4.2 BSD ならさらに使いやすかったであろう）をＶＡＸ１１で用いた。ファィルシステムは，木構造であり，４．２節で述べるドキュメンテーション方式の基礎となった。

マルチウィンドウエディタＥｍａｃｓは，「拡張可能で改造可能な自己文書化されている」エディタとして設計されたＭＩＴのエディタである。（ＣＭＵのＪ．Gosling の　Unix 版を使用した。〔７〕）この高い拡張可能性をいかして種々の機能を付加した。しかし，文字列（テキスト）を扱うエディタという性格上，マルチウィンドウ・マルチバッファであってもマルチフォントやイメージを扱うことは当然困難である。すなわち，ビットマップディスプレイの意義が薄れるということである。

ビットマップディスプレイは，米ＢＢＮ社製のＢitGraph TM である。（７６８×１０２４ピクセル，８５字×６４行）縦長ディスプレイで行数が多いものが好ましい。もっとも回線速度が遅いと画面制御に時間がかかるのでよくない（9600ボー程度が適当）。

マウスは，上記ディスプレイにオプションのもので３ボタンのものである。Emacsはその拡張性のためマウスを簡単にサポートすることができる。この組合せによりＥmacsの操作性は格段に向上した。（キー入力したいウィンドウの好きな位置へマウスの指示を合わせ，キーボードをたたくのは，まさに入力される文字列だけである。）

ウィンドウシステムはオーバーラッピング可能なものを作成して使用した（図12）。これにより通常のエディタの vi や図形なども扱える。

4．2　ドキュメンテーション方式

図11に示すように，トップダウン設計によって作られた各モジュールごとに，仕様書，ソースコード，マニュアル，テストデータなどを用意する。これらを統一的に取り扱うことにより，ドキュメント群の管理を容易にしている。すなわち，モジュール別，版別，種別（表１）によって細分化したドキュメントファィルを作る。補助ファイルである“Table”，“Where” を参照することにより，

・プロダクト全体の概説書やマニュアル
・あるレベルまでの仕様書
・あるモジュールに関するすべての文書
・通常のソースリスト
・プロダクトの設計メモ

等を取り出したり新たに作成することが可能である。さらに，これらの操作を過去の版について行うこともできる。

ドキュメント群を操作する各種ツール群は，Ｃ－Shell Script （コマンドランゲージ）で記述作成した。（ラピッドプロトタイピングである。）ただし，効率が悪い（時間がかかる）ため，仕様が確定しだい順次言語Ｃで書き直す予定である。このようにMOSDENV　を構成する各種ツール群自体も自己文書化・段階的詳細化しかもラピッドプロトタイピングというMOSDENV　の基本方針に基づいて作成された。

表２～６は，通常のドキュメント（ソースコード）とMOSDENV　によるドキュメント群を比較したものである。簡単のため行数のみによる比較にとどめた。表２，４，６よりMOSDENV　では約２倍の量のドキュメントを必要とすることがわかる。また表５より第１版から第２版への改版にともない約２割の記述量の増加がみられる。さらに，表４，６から段階的詳細化のあとがうかがわれる。

```
overview : rough sketch of the module
description : functional description of the module
spec      : specification of the module
memo      : design memo of the module
source    : source program of the module (source.c)
pcl       : program check list (test data) of the module
interface: condition of the module as a blackbox
manual    : manual of the module, usage description
(manual.roff)
object    : compiled object (source.o)
Table     : elements of the lower modules
Where     : elements of the upper modules
History   : check of the time-dependency
```

Table 1. Attributes of a document (in Document Formula)

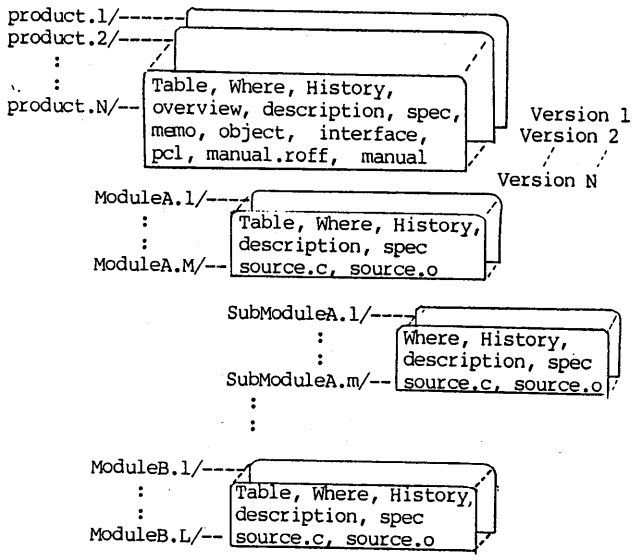


Fig. 11　An Example of the Document Formula

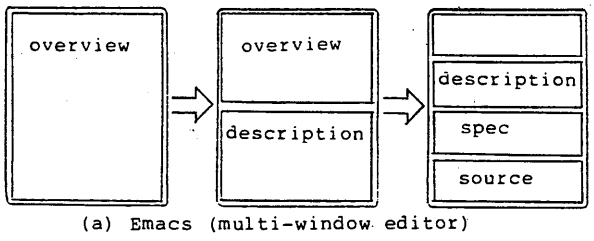
4．3　プロンプティングシステム

MOSDENV のドキュメント群を効率よく作成してゆくためには，プロンプティングシステムが必要である。これは，ユーザにドキュメントを書く順番・内容・構成等を促すシステムである。図14に示す順序で作成を進行する際にマルチウインドウエディタが有効に働く。

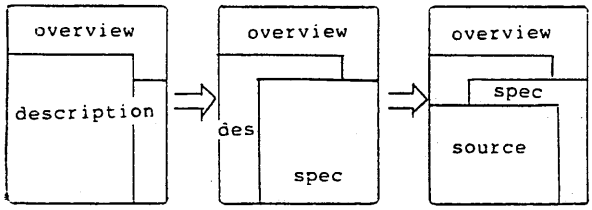
プロンプティングシステムの概念構成は図13のように「演劇」と対比することができる。良い役者（シナリオインタープリタ）だけでなく良い台本・原作も必要であることがわかる。具体的には，シナリオインタープリタ（ＳＩ），プログレスチェッカ（ＰＣ）が，各々プロセスとして存在し，ＳＩはシナリオをユーザに伝え，ＰＣはドキュメント群の生成状態を観測する。要求されることとして―

・シナリオの動的変更可能性
・ユーザ設定シナリオ（プロファイル）
・ユーザレベル（システムに対する慣れ具合）
・タイマーによる進行状況の把握
・メニュー（今何ができるのか）
・問い合わせ（今何をすべきなのか）
・ｄａｅｍｏｎとａｎｇｅｌ（図14の関係に従ってコンパイル等を実行したり，ユーザに告げたりするＰＣのサブプロセス）

などがある。



(a) Emacs (multi-window editor)



(b) Overlaying Multi-window Editor

Fig. 12　Stepwise refinement with multi-window editor

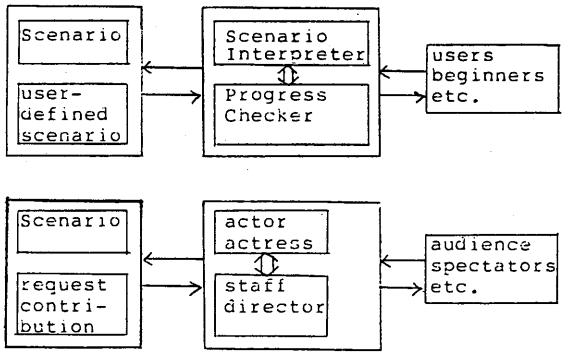


Fig. 13 Conceptional Model of the prompting-system and a comparison with a play

Table 2　NORMAL window_demo/

	file	line	word	byte
source	1	159	306	3577
object	1	-	-	59727
aux.	10	-	-	16213

note: aux. -> image file etc.

Table 3　MOSDENV window_demo/

	dir	file	line	word	byte
*	4	27	-	-	-

note:
"object and aux." of MOSDENV is as same as NORMAL.
They are in window_demo.1,　"source" of NORMAL is extended.

	window demo	sigdemo	stlpasswd
NORMAL(source)	159	430	87
MOSDENV(total)	322	625	182
(ratio)	2.0	1.5	2.0
overview	8	4	1
description	31	45	19
spec	102	187	42
source	160	387	87
others	22	2	33

Table 6　A comparison with NORMAL document and MOSDENV's Document Formula

Table 4　MOSDENV window_demo/　　number of line

	o.v.	des.	spec	src	i.f.	man.	memo	(total)
window_demo.1	-	-	-	-	8	-	-	8
main.1	5	15	57	103	-	2	11	193
inthand.1	2	5	24	21	-	-	-	52
tekbox.1	1	11	21	36	-	-	-	69
(total)	8	31	102	160	8	2	11	322

note: o.v. -> overview, des. -> description, src -> source,
i.f. -> interface, man. -> manual

Table 5　after revised;
MOSDENV　　window_demo/　　number of line

	o.v.	des.	spec	src	i.f.	man.	memo	(total)
window_demo.1	-	-	-	-	8	-	-	8
window_demo.2	-	-	-	-	8	-	-	8
main.1	5	15	57	103	-	2	11	193
main.2	6	18	65	125	-	2	11	227

note: _____/
ratio -> about 20 % up

注)
プログラムの説明
“windowdemo”，
“sigdemo”はBitGraph用の言語Ｃで書かれたデモプログラムである。
multi-windows，
image displayer etc.
これらを MOSDENV に実際にあてはめてみたものが表2～6の結果である。

4．4　改版支援ツール

リリースするまでは，4．3節で述べたプロンプティングシステムに従いドキュメントを作成してゆく。リリース後は，あくまでもトップダウンを方針とする―

▷ひとつのモジュールの中での変更の場合

図14の順序に従ってのみ変更される。

(1)上位ドキュメントが変更されれば下位ドキュメントは必ず変更（再読）されなければならない。

(2)あるドキュメントを変更したいときは，その上位ドキュメントを他のウィンドウで参照しながら，その意図の範囲内で変更しなければならない。

▷複数モジュールの改版の場合　　(Main → Sub)

メインモジュールの変更の結果，ドキュメントのTable, interface が変更されれば，該当するサブモジュールを変更する（サブモジュールの“overview”の変更を最初に促すことになる。）

などである。これは，ドキュメント群間の構造を保ちつつ編集ができるという点で広い意味での構造エディタといえよう。Unix には“MAKE”というメンテナンス支援のツールがある。MAKEはたいへん有用で，対象が多いとき，処理手順が複雑なときは特に役に立つ。しかし，あくまでも作業手順が前もって明確にされていなければならない。たとえば，コンパイル，リンクなど。一方，この「改版支援ツール」は広汎なドキュメント群の改版と管理とをねらったもので処理手順を動的に変更できる特徴をもつ。

4．5　評価と検討

マルチウィンドウエディタを利用したドキュメント群の管理とソフトウェア開発をねらったMOSDENV　の今回の実験システムから次のようなことがわかった。

①ドキュメントの細分化・階層化にともなう量の増加は，通常のソースプログラムと比較して約2倍である。

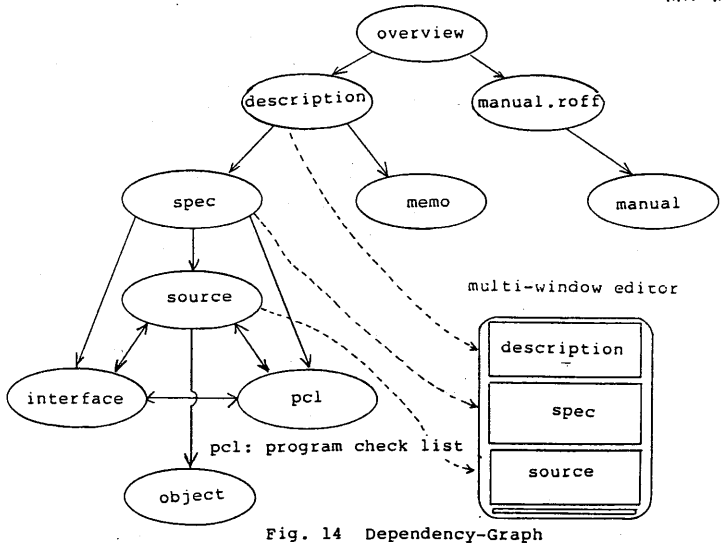


Fig. 14　Dependency-Graph

②トップダウン設計・モジュール分割によるソフトウェア開発に，C-Shell Script やawk などの実働可能なものを素早く作成できるツール群を加味することにより，ラピッドプロトタイピングの要素を盛り込むことができる。

③階層化されたドキュメントと木構造ファイル，さらにマルチウィンドウエディタを組み合せることによってソフトウェア開発におけるユーザインターフェースの向上がはかられる。（現在，支援ツール群の整備中）

④マルチウィンドウシステムについて，個々のウィンドウがそれぞれ重なり合うもの（当研究室）と重なり合わないもの（Emacs ）との比較をさらに行うことが必要である。（図12の(a)と(b)）

⑤プロンプティングシステム・改版支援ツールに代表される支援ツール群が充実されなければ，システム全体の機能向上は得られない。

⑥ソースプログラムの readability, understandability は，ドキュメンテーション方式（とそのツール群）によって向上した。

⑦保守用データベースについては今回は考察しなかった。

機能向上・仕様変更がひんぱんに行われるソフトウェア開発時のソフトウェア保守という問題を採り上げた。今後，ドキュメンテーション以外の開発環境の問題についても個々に採り上げる必要があろう。

5．おわりに

ドキュメントのオンライン化にともないソフトウェア開発の方法論も環境も変わりつつある。一方，保守しなければならないソフトウェアは山積している。今後，ソフトウェア開発時点において保守を前もって考慮しておくことがますます肝要であろう。

MOSDENVは，　開発途中であるが，今後その試作品，評価ができしだい順次発表してゆきたい。

参考文献

〔1〕長田　他：「MOSDENVにおけるドキュメンテーション方式」
情報処理学会第27回全国大会，1983
pp619－620

〔2〕長田　他：「MOSDENVの実装とその評価」(予定)
情報処理学会第28回全国大会，1984

〔3〕「ソフトウェアエンジニアリングに関する調査研究『プロトタイピング技法』」，
ソフトウェア産業振興協会，1983

〔4〕D.D. MaCracken, M.A. Jackson,
“Life Cy-cle Considerd Harmful,” SEN Vol.7 (2) pp.29-32 ACM SIGSOFT , (Apr. 1982)

〔5〕G. R. Gladden,“STOP THE LIFE-CYCLE, I WANT TO GET OFF,” SEN Vol.7 (2)　pp.35-39 ACM SIGSOFT , (Apr. 1982)

〔6〕ACM SIGSOFT,“SPECIAL ISSUE OF RAPID PROTOTYPING,” SEN Vol.7 (5) (Dec 1982)

〔7〕James Gosling,“UNIX EMACS” CMU (May 1982)