バージョン管理ファイルシステムを用いた保守支援ツールの提案

山本 哲男[†]　松下 誠[†]　井上 克郎[†‡]

[†] 大阪大学大学院基礎工学研究科
〒560-8531 大阪府豊中市待兼山町 1-3

[‡] 奈良先端科学技術大学院大学情報科学研究科
〒630-0101 奈良県生駒市高山町 8916-5

Email: {t-yamamt,matusita,inoue}@ics.es.osaka-u.ac.jp

近年のソフトウェアはより大規模化しており，より複雑になってきている．また，ネットワークを用いた分散開発も一般的になってきている．このような開発環境においては，ソフトウェアの開発や保守作業はより困難なものとなってきている．本研究では，ソフトウェアの開発における保守作業を支援するための開発環境 Moraine の提案を行う．Moraine では開発環境自身の変更履歴をバージョン管理ファイルシステムを用いて蓄積し，保守作業の際に利用することが出来る．また，Moraine のデータ蓄積部分に関して実装と評価を行い，有効性の確認を行った．

ソフトウェア開発環境，バージョン管理

Software Maintenance Environment with Version Control File System

Tetsuo Yamamoto[†]　Makoto Matsushita[†]　Katsuro Inoue[†‡]

Graduate School of Engineering Science,
Osaka University
1-3 Machikaneyama, Toyonaka,
Osaka 560-8531, Japan

Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma,
Nara 630-0101, Japan

Email: {t-yamamt,matusita,inoue}@ics.es.osaka-u.ac.jp

Recently software is large-scale and complex, and is being developed in more distributed way using a network. In these circumstances, it is more difficult to develop and maintain the software. In this paper, we propose Moraine, an accumulative software development environment for supporting a software maintenance. Moraine records all the changes of files as an evolution history, and trace a history by specified date or query from users' request. We also implement the data accumulative function of Moraine, and show the performance of file recording.

Software Development Environment, Version Management

1 まえがき

近年のソフトウェアは大規模化しており，より複雑になってきている．また，開発形態も大人数での開発や分散化の傾向にある．このような環境での，ソフトウェア開発やソフトウェアの保守作業というのは非常に難しいものとなっている．そこで，ソフトウェアの品質や保守性を向上させるための研究が数多く行われている．しかしながら，現在のソフトウェアは新しい環境で実行する際には，その新しい環境に合わせてソフトウェアを変更しなければならない．このソフトウェアの保守にかかるコストは非常に大きなものとなっている．これに対する解決策は，新しい環境になったとしてもソフトウェア自身が柔軟に対応するようにソフトウェアを開発することである．しかしながら，我々はソフトウェア自身だけでなくソフトウェア開発環境自身も環境の進化に対応すべきであると考えている．

将来のソフトウェア開発環境は過去や現在に行った開発環境の拡張であるため，ソフトウェアの進化に対応するための一つの方法としてソフトウェア構成管理やバージョン管理を行う方法がある．ソフトウェア構成管理はソフトウェア開発過程で作成されるプロダクトの識別や制御，状態の把握等をを解決する研究である[3]．ソフトウェア構成管理は開発過程の各段階で行なわれる．設計段階では構成要素の決定や確認，コーディング段階では生成物の変更状態の追跡，リリース段階ではリリースされる生成物の特定を，それぞれ効率良く行なうことを目的とする．しかしながら，ソフトウェアの一貫性を保った構成管理は容易ではない．

多くの場合，RCS[11]やCVS[1]といった既存の構成管理ツールを使用している．しかし，これらの構成管理ツールでは使用するためには，ある特定の方法を正しく用いなければならない．そのため，これらのツールを始めて使う場合には教育や訓練が必要である．また，これらのツールはある特定の構成管理モデルに依存しており，変更することは容易ではない．

本研究では，ソフトウェアの進化を支援する新しいソフトウェア開発環境Moraine を提案する．Moraineでは，バージョン管理を行うファイルシステムを用いることによりソフトウェア開発環境そのものを蓄積することができる．Moraine による開発では，ソフトウェアの開発中に生成したすべてのファイルに対する変更が自動的に記録され，いつでも任意の時点のファイルを取り出せることが可能となる．

Moraineの考えの基礎には以下の要因がある．

- ディスクやCPUに要する費用が減少している．
  4章に示すように，ソフトウェア開発に必要な中間プロダクトの各バージョンをそのまま，又は圧縮して保存しても，通常，安価で市販されているディスクの容量に収まることが期待される．一人のソフトウェア開発者のために数十Gバイト程度のディスクを用意することが可能になりつつある．また，圧縮して保存するためのCPU負荷上昇も，4章に示すように，問題とはならないと考えられる．今後，より高性能のCPUを用いることにより，これらの負荷は相対的に減少すると考えられる．

- ありのままを開発環境を記録したい．
  プログラムの開発過程において，過去のバージョンのプロダクトや，過去に行なった操作をたどることが出来ると，開発管理や進捗情報の収集に都合が良い．例えば，プロダクトメトリクスやプロセスメトリクスについては，過去から今までのものを容易に収集することができ，プロジェクトの制御に非常に有効である．プロジェクトの途中で，メトリクス収集の方針を変更しても，過去にさかのぼって収集可能である．

- 管理のための諸作業を開発者から隠蔽したい．
  バージョン管理システムの導入のために後で述べるような特殊なシステムのインストールやその使用方法などを学ぶのは容易ではない．また，プロジェクト管理のために種々のメトリクス収集ツールの導入やその使用なども大変である．システムの導入や使用のための障害を少なくし，管理のための諸作業が陽に表れないシステムが開発者にとっては望ましいと思われる．

Moraine は大きく蓄積と参照という二つの機能から構成される．蓄積とは，ソフトウェアの開発においてファイルに関するすべての操作を自動的に記録する事である．この蓄積は過去の開発状況を示す．参照とは，この蓄積されたファイルを参照し分析することであり，過去に行われた作業の確認やソフトウェア開発の改善に用いることができる．

以降，2章では，Moraineの基本的な概念について述べ，3章では，Moraineで必要となる蓄積部分のシステムの実装について示す．そのシステムの評価と考察について4章で述べ，最後に，本研究のまとめと今後の課題を5章で述べる．

2 Moraineの概念

この章では，Moraineの二つの大きな特徴について述べる．

2.1 蓄積

Moraineは全てのファイルの変更を履歴として記録する．ユーザはファイルの保存に注意する必要はない．もし，ファイルが必要なく消去したいときは，いつでも現在の環境からファイルを消すことが出来る．Moraine

はファイルの消去を監視し，履歴としてはファイルを残しておき，ユーザにはファイルの存在を見せないように振舞う．ユーザは単にファイルを作成したり更新したり消去したりするだけで，Moraineはファイルの新しいバージョンを自動的に作る．そのため，ユーザは新しいバージョンを作成するための特別なことをする必要は無い．最初に，ツールの使い方などを学ぶ必要は無い．ファイルに対する読み書きなどの一般的な操作は，すべてMoraineによって自動的に記録される．ユーザはファイルについて何も管理する必要はない．結果として，通常のファイルシステム上のファイルの操作とMoraineで管理されているファイルの操作はユーザの視点からすると違いはない．

2.2 参照

Moraineで蓄積されたファイル履歴は，任意の時間やバージョン番号やあるバージョンにユーザによって付けられた名前を指定することで参照することが出来る．Moraineでは自動的に新しいバージョンが出来るので，ユーザにとって意味のあるバージョンに対しては名前を付けることが出来る．この名前はバージョンの別名として使用可能である．一つ以上のファイルに同じ名前を付けることで，それらのファイルを同時に参照することが出来る．また，ユーザはバージョン間の差分やブランチ分けといった操作も用意に行うこととが出来る．あるバージョンを指定することで，そのバージョンの時点の作業から再開出来る．

Moraineで蓄積されたファイル履歴はソフトウェアメトリクスに利用出来る．ソフトウェアメトリクスはプロジェクト管理に使われ，将来のプロジェクトの品質向上やコスト削減に非常に重要である．

3 Moraineの実装

本章では，Moraineの実装について述べる．Moraineは2章で述べた大きく二つの部分から構成されるが，今回はそのうち蓄積部分について実装を行った．今回は蓄積部分について実装を行った．蓄積部分はファイルシステム上のファイル全ての変更記録を保存するシステムであり，以降VCFSと呼ぶ．

3.1 VCFSの設計方針

VCFSの設計方針は以下の3つである．

- ファイルシステムとして実装することで，システムの操作を事前に学ぶ必要がない．
- ファイルシステムがもつファイル構造が専用の形式に固定されるという欠点を解消するために，既存のファイルシステムを用いた堆積型(stackable)ファイルシステムとして実装する[5]．
- ファイルシステムが用いるバージョン管理方式をファイルシステムから独立させることにより，複数の異なる管理方式を使い分けられるようにする．

3.2 VCFSの概要

VCFSでは，通常のファイルに対する読み出し(readシステムコール)，書き込み(writeシステムコール)動作を直接バージョン管理作業に対応付ける．そうすることで，特別な作業を伴わず既存のエディタ等によるファイルの読み書き作業だけでバージョン管理機能が利用できる．つまり，実際のファイル操作は，ファイルを操作するユーザプロセスにとっては一般のファイルシステム上のファイルを操作するのと同様の操作で可能である．

通常ファイルのみバージョンを取り，ディレクトリ，シンボリックリンク，特殊デバイスファイル，ソケット，名前付きパイプなどはバージョンを取らない．バージョンを取るのは，新規にファイルを作成した場合や，既存のファイルを変更した場合であり，それらを新規バージョンの登録作業として扱う．また，ファイルを読み出す場合，最新のバージョンを読み出すことになる．

通常，バージョン記録ファイルはファイルシステムからは操作を行わないが，ファイル名変更(renameシステムコール)，削除(removeシステムコール)についてはバージョン記録ファイルに対しても同様の操作を自動的に実行する．また，ファイルのcheckin作業が終わるまで，他のプロセスからはそのファイルに対して読み出しのみが行なえる．

3.3 VCFSの設計

VCFSが採用するバージョン管理手法はRCSなどで用いられているcheckin/checkoutモデル[4]を採用した．これは，バージョン管理を特定の方式に依存するのではなくcheckin/checkoutモデルに基づいたものであればそれが使用可能な事を意味する．今回の実装においては，RCSとVCSという二種類の管理方法を提供する．

VCFSはファイルシステム部分とそれに付随する管理用コマンド群，バージョン管理デーモン，バージョン管理サブシステムによって構成される．図1は各構成要素のつながりを表したものである．

VCFSのマウント元のファイルシステムには最新バージョンのファイルとバージョン記録ファイルを保持している．マウント先では最新バージョンのファイルのみ操作でき，バージョン記録ファイルは見ることが出来ない．例として，/versiondbを/projにマウントし，/proj/fooというファイルを作成した場合，/versiondbには，そのファイルfooとfooのバージョン記録ファイルが作成される．この際，/versiondbには通常ファイルは,aをバージョン記録ファイルには,vをファイル名の最後に

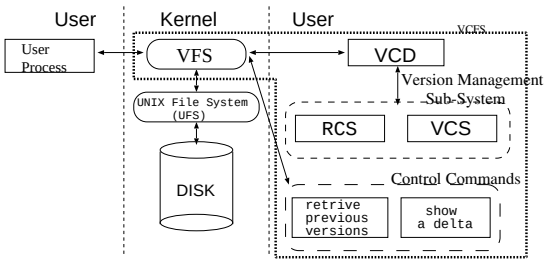

図 1: VCFS の構成

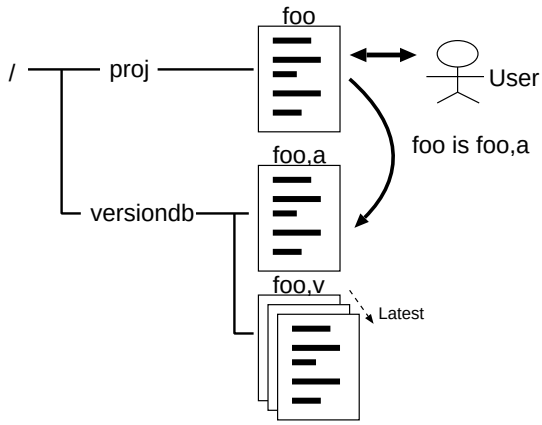

図 2: ファイルシステム上のファイル

付加したファイル名として保存される．図2に示すように/proj/foo は /versiondb/foo,a というファイルを指すことになる．また，/versiondb/foo,v は /proj の下では見ることが出来ない．ユーザは /proj の下のファイルのみを操作することになる．

常に最新バージョンのファイルを 1 つのファイルとしてファイルシステム上に置いておく．上の例では，/versiondb/foo,a になる．そのため，ファイルの読み書きはそのファイルに対する読み書きになるため高速に動作する．ファイルを読み出し専用で開いた場合は，従来のファイルシステム上と同じ動作をする．それに対し，書き込みフラグを付けてファイルを開いた場合，読み出し，書き込みといった操作は従来と同じであるが，ファイルに対して閉じる動作(close システムコール)を行ったとき，そのファイルに対して checkin 動作を行う．この checkin の動作はカーネル内では動作せず，カーネルの外のバージョン管理デーモンとバージョン管理サブシステムが行う．

バージョン管理機能をカーネル内に記述しない利点としては，checkin/checkout の動作をカーネルを変更せずに切替えが出来る事があげられる．また，RCS 等のプログラムを使ってバージョン管理を行っていた場合，RCS で使われている機能をバージョン管理サブシステムのプログラムとして記述することで，バージョン管理システムの移行が容易となる．また，checkin/checkout 用のツールを，独自の形式で書くことも可能であり，バージョンを記録ファイルに必要な情報を付加したバージョンを付加することが出来る．

3.3.1 ファイルシステム VFS

ファイルシステムとしての機能をもつ部分であり，カーネル内のプログラムである．このファイルシステムに対してユーザプロセスからファイルの読み出し，書き込み，ディレクトリの作成等といった一般のファイルシステムの持つ機能を行うだけで，バージョン管理機能が動作する．

ファイルシステムには直接物理装置を操作する UFS[10] や LFS[2] といったものや，ネットワークを用いた NFS といったものがある．VCFS で用いられるファイルシステムは，既存のファイルシステムの上に構築する堆積型ファイルシステム [5] として設計した．

堆積型ファイルシステムは，既存のファイルシステム内部の変更は行わない．また，ファイルシステムの出力は HDD 等に直接行われず，下層となるファイルシステム上のファイルとして行われる．記憶装置やネットワークを処理するコードを書く必要ががなく，移植が容易である．

3.3.2 バージョン管理デーモン VCD

カーネル内のファイルシステムとのやりとりを行うデーモンプログラムである．カーネルから依頼されるファイルのバージョン管理を引き受ける．ファイルに対する実際のバージョン管理は，バージョン管理サブシステムに依頼する．

3.3.3 バージョン管理サブシステム

実際にバージョンを記録する部分である．バージョン管理サブシステムは複数のバージョン管理ツールの集合体である．バージョンを記録を行う場合，いずれかが選択されてそのツールを実行する．

VCFS では，バージョン管理サブシステムとして，既存のバージョン管理システムを用いており，ファイルシステムを利用する際に，どのシステムを利用するかが選択できる．今回の実装では RCS と VCS の 2 種類のバージョン管理サブシステムが利用可能である．

RCS RCS は各バージョンの差分のみを保存するために広く用いられているツールである．RCS ではバージョンの派生 (木構造) を許す．この派生された枝をブランチと呼ぶ．

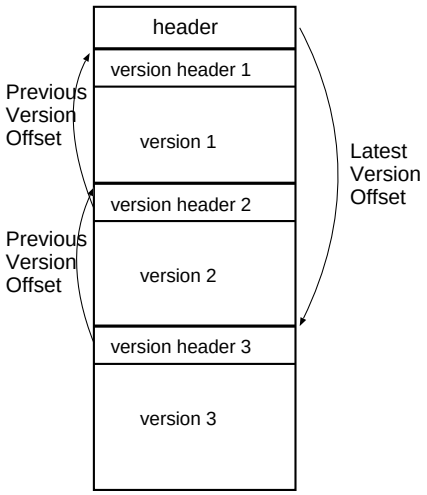

図 3: VCS のファイル形式

表 1: ヘッダ構造

名前	サイズ	内容
バージョン番号	4 バイト	最新バージョン番号を表す．初期バージョンを 1 として以降 2，3，... と続く．
最新オフセット	16 バイト	最新バージョンのヘッダのオフセットを表す．

このサブシステムではバージョン記録ファイルに対する操作はすべて RCS のコマンドを内部で実行することでバージョンを記録する．

VCS 今回作成した単純な機能を持つバージョン管理システム．各バージョンを圧縮しない形でバージョン間の差分もとらずにそのまま保存する．バージョンの派生はなく，線型である．RCS と違いバージョン間の差分情報をとらないことで高速に動作させることを目的としている．

VCS のバージョン履歴ファイルのファイル形式を図 3にファイルのヘッダを表 1にバージョンヘッダを表 2に示す．checkin 動作は登録するバージョンをバージョンヘッダと共にバージョン履歴ファイルに追加していく．ただし，ヘッダのバージョン番号と最新オフセットは更新しておく．過去のバージョンはヘッダの最新オフセットと各バージョンファイルのバージョンヘッダの前オフセットを辿ることで取得可能となる．バージョン記録する際の情報として，ファイルの属性をすべて保存するため過去のファイルを取り出すと，完全に過去の状態になる．

表 2: バージョンヘッダ構造

名前	サイズ	内容
属性	vattr 構造体のサイズ	ファイルの属性を表す vattr 構造体．
前オフセット	16 バイト	前バージョンのヘッダのオフセットを表す．

3.3.4 管理用コマンド群

通常のファイル操作では行えない操作をするためのコマンド群である．

コマンドとしては以下のものがある．

- 任意のバージョンファイルの取り出し
  リビジョン番号や日付などを指定し，過去のバージョンファイルを取り出すコマンドである．
- ブランチ作成
  RCS などバージョンの派生を許すモデルの場合に，派生を行うコマンドである．VCS など許さない場合はこのコマンドを実行しても何も起こらない．
- 差分の閲覧
  バージョン間の差分を取り出すコマンドである．

3.4 VCFS の実装

BSD UNIX[8, 9] 系のオペレーティングシステムである FreeBSD 3.0-RELEASE [6] を対象にして実装を行った．

実装はすべて C 言語で記述し，全体で 5000 行程度になった．それらの内訳は，カーネルへの追加としてのファイルシステムに 4500 行，バージョン管理デーモンに 340 行，その他のバージョン管理サブシステムとのインタフェース等に 300 行である．

4 評価

バージョン管理ファイルシステムの評価を行う．ファイルシステムとしての実用性を考えるうえで有効かどうかを調べる．評価の方法としては，ファイルシステムとして通常用いられる UNIX ファイルシステム (UFS) と堆積型ファイルシステムの一つである NULLFS との比較することで行った．

ファイルシステムの評価として以下の 2 種類のテストを行った．

- 様々な大きさのファイルの読み出し，書き込みのようなファイルシステムの性能に関するもの．

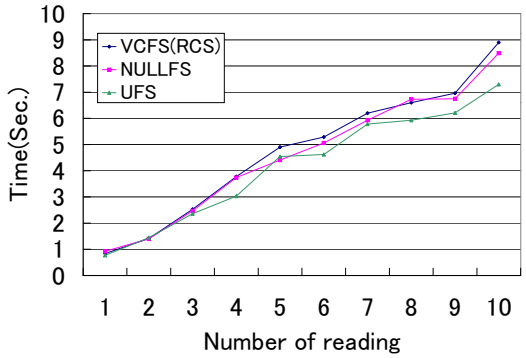

図 4: 1MB の連続読み出し時間 (Sec)

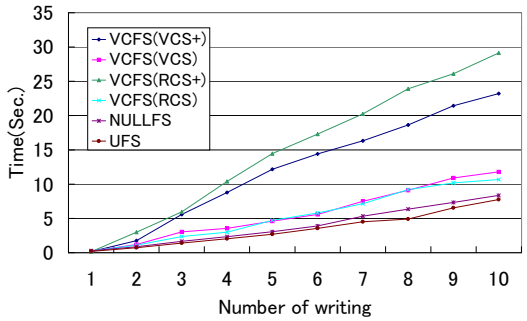

図 5: 1MB の連続書き込み時間 (Sec)

- ファイルシステム上に蓄えられるファイルの大きさに関するもの．

以下で述べるテストはすべて Pentium 166MHz・48MB RAM の計算機で評価した．

4.1 性能テスト

4.1.1 読み出しテスト

ファイルの大きさが 1MB の異なるファイルを単一プロセスで繰り返し連続して読み出す時の処理時間を計測した．ここで処理時間とは連続した読み出しを行うプロセスの実行時間である．つまり，プロセスの生成から消滅までの時間となる．読み出すファイルは既にファイルシステム上に読み出す回数分だけ用意しておく．1 回だけの読み出しから 10 回連続までの 10 個の値が各ファイルシステムについて存在する．

測定結果を図 4に示す．VCFS(RCS) と VCFS(VCS) はそれぞれバージョン管理サブシステムとして RCS と VCS を用いたという事を意味する．

どの回数においても VCFS(RCS) と NULLFS はほとんど違いが見られない．これによりファイルの読み出しに関してはファイルシステム内のバージョン管理機能はほとんど時間がかからないことが分かる．VCFS(RCS) や NULLFS は UFS と比べると約 10%余計に時間を要するがこれは堆積型ファイルシステムとして実装したためだと考えられる．

4.1.2 書き込みテスト

同様のテストを，書き込みに関しても行った．ファイルの大きさを 1MB とし，異なるファイル名で単一プロセスで繰り返し連続して書き込む時の処理時間の計測を行った．ここで処理時間とは連続した書き込みを行うプロセスの実行時間である．書き込むファイルはファイルシステム上に存在していないものとする．1 回だけの書き込みから 10 回連続までの 10 個の値が各ファイルシステムについて存在する．

測定結果を図 5に示す．VCS(SYNC) はバージョン管理サブシステムとして VCS を用い，完全にバージョンを記録するまでの時間を計測したものである．RCS(SYNC) についても同様である．

UFS と比べ，NULLFS は約 10%余計に時間がかかる．これは読み出しの際の UFS と NULLFS との違いと同じである．読み書き共に 10%程度の時間を余計に必要とする．VCFS と UFS 比べると約 30%の VCFS のほうが遅い．読み込み時には 10%であったが，書き込みの際にはバージョンを保存する時間を要するためである．しかし，バージョンの記録が完全に終了する時間は倍以上の時間を必要とする．今回の実装ではバージョンの記録はカーネル内では行わず，ユーザプロセスとして実行する．そのため，30%程度の時間で書き込みを行うプロセスは終了し，バックグランドでバージョン管理サブシステムが動作しバージョンの記録を行う．バージョンを記録する機能をカーネル内部から外したことで高速性をもつファイルシステムが実現出来た．

また，ファイルの大きさが 32KB にして書き込みテストを行った．測定結果を図 6に示す．

基本的に上記の 1MB ファイルの書き込みと同じ傾向が見られるが，ファイルの大きさが小さいときには，バージョン管理サブシステムとして RCS より VCS を用いたほうが高速に動作する．

4.1.3 バージョン記録テスト

ファイルの更新時の時間について測定を行った．ファイルの大きさが 1MB のファイルを用意し，各ファイルは RCS で差分が最大なるようにした．更新回数が 0 というのは新規にファイルを作成したという事である．測定結果を図 7に示す．

バージョン管理サブシステムとして RCS を用いた場

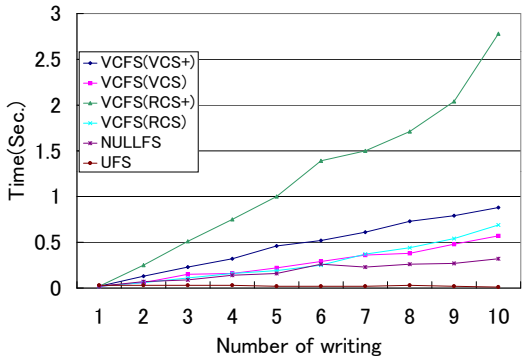

図 6: 32KB の連続書き込み時間 (Sec)

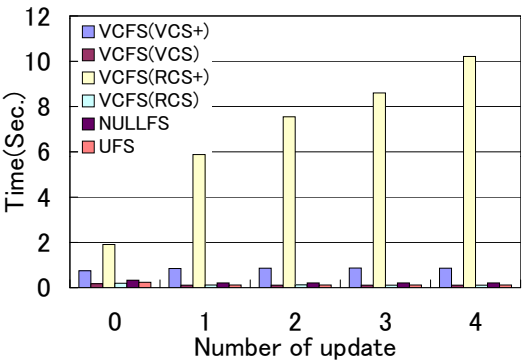

図 7: バージョンの更新による時間 (Sec)

合を除いて更新したときに要する時間は新規にファイルを作成した時間と同じである．バージョン管理サブシステム VCS は以前のバージョンからの差分は取らずにバージョンの記録を行うため更新回数によらずファイルの大きさが同じならば，更新する時間はほぼ一定である．RCS の場合，前バージョンとの差分をとるため更新回数が増えると，更新する時間も増加する．

4.1.4 コンパイルテスト

実際のアプリケーションのコンパイルに要する処理時間を測定した．測定対象となるアプリケーションとしては FreeBSD に付属の tar と dump のプログラムとした．アプリケーションのソースファイルは各ファイルシステム上に用意しておき，make の開始から終了までを処理時間とした．測定結果を表 3に示す．

make が行う作業は，異なるソースファイルのコンパイルの繰り返しである．コンパイルはソースファイルの読み出し，そのソースファイルから生成されたオブジェクトファイルの書き込みであり，ファイルシステムに対して読み書きを行う．VCFS は UFS 比べると 20%程度の時間を要する．これは，実用上問題ない程度の時間

表 3: アプリケーション作成時間 (Sec)

	dump	tar
VCFS(VCS)	10.96	28.63
VCFS(RCS)	12.79	33.46
NULLFS	11.3	32.01
UFS	10.9	28.27

表 4: データ情報

	最終ソースプログラムの行数	最終ファイル数	総バージョン数	cc 成功回数
data1	9339	45	311	222
data2	4067	20	147	92
data3	2543	18	247	110

である．

4.2 容量テスト

大阪大学基礎工学部情報工学科で行われるコンパイラ作成演習で得られたプログラムの編集履歴データを VCFS に適用した．ファイルを時刻順にファイルシステム上にコピーしていく．また，更新されたファイルについてはコンパイル (cc) を実行する．適用したデータを表 4に示す．

各ファイルシステム上でのソースファイルとコンパイルした結果のオブジェクトファイルも含む最終的なファイルの総容量を表 5に示す．

UFS 上の容量が元の大きさであり，この大きさと比べるとバージョン管理サブシステムでの容量は大きくなるが，RCS を用いた場合は数倍程度であり，VCS を用いた場合は数十倍まで増加する．採用するバージョン管理サブシステムによってディスク使用容量が大きく変化する．

4.3 比較

既存のバージョン管理システムと VCFS についての比較を行う．

表 5: ディスク使用容量 (KB)

	UFS	VCFS(RCS)	VCFS(VCS)
data1	225	1388	3149
data2	117	546	1377
data3	73	604	1501

- RCS とVCFS
  RCS はツール群により構成されたバージョン管理システムであり，VCFS はファイルシステムにバージョン管理機能をもつバージョン管理システムである．しかし，VCFS では RCS を内部で用いることで RCS が持つ機能をそのまま利用できる．

- 3D Filesystem とVCFS
  3D Filesystem は UNIX SystemV Release 3 のカーネルを変更することによって，ファイルシステム上に実装されたバージョン管理システムである [7]．3D Filesystem では RCS と同様，ファイルシステム上のファイルをプロダクトとして扱っている．3D Filesystem，VCFS ともにファイルシステムにバージョン管理機能をもつバージョン管理システムである．ただし，3D Filesystem では checkin をユーザが明示的に指示する必要がある．

4.4 考察

性能面およびディスク容量の観点から評価を行った．今回の評価から以下のようなことが分かる．

- バージョン管理ファイルシステムは一般のファイルシステムに比べ 20%程度遅いが実用上問題ない．

- バージョン管理サブシステムとして RCS ををを用いれば，ディスク使用容量は 10 倍程度であり実用上問題ない．

- 高速性を求めるならば VCS を，ディスク使用容量を減らしたいならば RCS をバージョン管理サブシステムに採用すれば良い．

5 まとめ

本研究では，ソフトウェアの進化を支援するソフトウェア開発環境に Moraine について述べ，Moraine の基本部分である蓄積部分の実装を行った．Moraine は，ソフトウェアの開発中に生成したすべてのファイルについて自動的にすべての変更を蓄積し，蓄積したファイルをいつでも取り出せることが可能となる．

今後，ユーザが容易に使用できる保守管理ツールのユーザインタフェースに関する考察を行い，さらに Moraine の実装を進める予定である．また，実運用による Moraine のユーザビリティの評価も必要だと考えられる．

参考文献

[1] Berliner, B.: CVS II: Parallelizing Software Development, *Proceedings of 1990 Winter USENIX Conference*, Washington, D.C. (1990).

[2] Carson, S. and Setia, S.: Optimal Write Batch Size in Log-Structured File Systems, *Computing Systems*, Vol. 7, No. 2, pp. 263–281 (1994).

[3] Conradi, R. and Westfechtel, B.: Version Models for Software Configuration Management, *ACM Computing Surveys*, Vol. 30, No. 2, pp. 232–280 (1998).

[4] Feiler, P. H.: Configuration Management Models in Commercial Environments, Technical Report CMU/SEI-91-TR-7, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213 (1991).

[5] Heidemann, J. and Popek, G.: File-System Development with Stackable Layers, *ACM Transactions on Computer Systems*, Vol. 12, No. 1, pp. 58–89 (1994).

[6] Hubbard, J. K.: RELEASE NOTES FreeBSD Release 3.0-RELEASE. This document is available on the World-Wide Web at the URL "http://www.freebsd.org/releases/3.0R/notes.html".

[7] Korn, D. G. and Krell, E.: A New Dimension for the Unix File System, *Software–Practice and Experience*, Vol. 20, No. S1, pp. 19–34 (1990).

[8] Leffler, S., McKusick, M., karels, M. and Quarterman, J.: *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley (1989).

[9] McKusick, M., Bostic, K., karels, M. and Quarterman, J.: *The Design and Implementation of the 4.4BSD UNIX Operating System*, Addison-Wesley (1996).

[10] McKusick, M., Joy, W., Leffler, S. and Fabry, R.: A Fast File System for UNIX, *ACM Transactions on Computer Systems*, Vol. 2, No. 3, pp. 181–197 (1984).

[11] Tichy, W. F.: RCS – A System for Version Control, *Software–Practice and Experience*, Vol. 15, No. 7, pp. 637–654 (1985).