U.D.C. 681.32.067

ライブラリ管理システム“LIME”

Library Management and Editing System ``LIME''

木村伊九夫* *Ikuo Kimura*
今居和男* *Kazuo Imai*
山東　滋* *Shigeru Santô*

プログラム開発の生産性は，ソースプログラムを管理するライブラリ管理システムの機能によって大きく左右される。

ソースプログラムを効率良く管理し，生産性向上を図ることができるライブラリ管理システムの出現が期待されてきた。

日立製作所はこのような要求にこたえるために，VOS 3 ライブラリ管理システム“LIME”を開発した。

LIMEは，プログラムの開発管理を容易にすることを目的とした新ライブラリ管理システムであり，次に述べるような特長がある。

(1) ファイル媒体のスペース効率を向上する。
(2) 複数のユーザーによりライブラリを同時に使用できる。
(3) プログラム開発管理のための管理情報が収集できる。
(4) 移行が容易である。

1 緒　言

電子計算機システムが広範に普及し，適用業務が拡大してゆくに従い，開発するプログラムの数及び規模は増大しつつある。

このような状況に対応するために，プログラム開発の生産性を向上するための施策が重視されてきた[1)]。プログラム開発の生産性を左右する重要な要因の一つとして，ソースプログラムを管理するライブラリ管理システムの機能を挙げることができる。

従来，ソースライブラリは，ソースプログラムの単なる入れ物(容器)として位置付けられてきた。しかし，プログラムの規模が増大するにつれて，そのスペース効率，操作性が重視され，同時に，プログラム開発管理のための情報を提供するなどの機能が要求されるようになってきた。

一方，TSS(Time Sharing System)の普及に伴い，プログラムの開発環境は大きく変化し，ライブラリ管理の方式も，TSSの利用形態に適合できるものであることが要求されるようになってきた。

ライブラリ管理システム“LIME”(Library Management and Editing System)は，ライブラリ管理でのこのような要求にこたえるために開発したライブラリ管理システムであり，ソースプログラム及びプログラムの開発管理情報を一体化して，集中的に効率良く管理することができる。

本稿では，以下，LIMEの開発背景，特長及び構成について紹介する。

2 開発の背景

VOS 3 (Virtual storage Operating System 3)システムでは，従来，ソースプログラムの管理には区分データセット※)が利用されてきた。しかし，この方式ではライブラリ構造などに起因する以下に述べるような限界があり，柔軟性のあるライブラリ管理の実現を妨げる要因の一つとなっていた。

(1) 無効領域が発生しやすく，スペース効率が悪い。
(2) 無効領域の詰め替えなどの保守作業が必要である。
(3) 複数のプログラマが同時に作業を進めることが困難である。
(4) プログラム開発管理のための管理情報が少ない。

特に，TSSの利用が普及してきた現在では，「複数のプログラマが，同時に作業を進めることができない。」という点は，非常に厳しい制約となっていた。

このような背景のもとで，ライブラリ管理上の諸問題を改善し，柔軟性のあるライブラリ管理を実現することによってプログラムの開発，保守の効率を向上するために開発されたのが，VOS 3 ライブラリ管理システム“LIME”である。

LIMEは，新形式のライブラリを導入することによって，ライブラリスペースの有効利用，プログラムの集中管理機能，プログラム開発管理情報の拡充などを実現し，効率の良いソースプログラムライブラリの管理を可能にしたプログラムプロダクトである。**図1**に，LIME開発の背景とLIMEの特長を示す。

3 LIMEの特長

3.1 ライブラリスペースの有効利用

プログラムの開発規模が増大するに従い，ソースプログラムが占めるライブラリスペースも増大する。LIMEでは，**図2**に示すライブラリ構造により，ライブラリスペースを有効に利用している。LIMEのライブラリスペースの利用方法には以下に述べるような特徴がある。

(1) 無効領域を発生させないライブラリ構造である。

メンバ更新後の旧メンバの領域は，無効領域とはならず未使用領域となり，再使用できる。

(2) 一つのメンバに対し，非連続な領域を割当てが可能である。

更新の繰返しにより，ライブラリ中の空き領域に断片が生

※) 区分データセット：一つのデータセット内に論理的なデータのまとまり(これをメンバと呼ぶ。)を複数個格納できるようにしたデータセットである。通常，ソースプログラムライブラリやロードモジュールライブラリとして使用されている。

* 日立製作所ソフトウェア工場

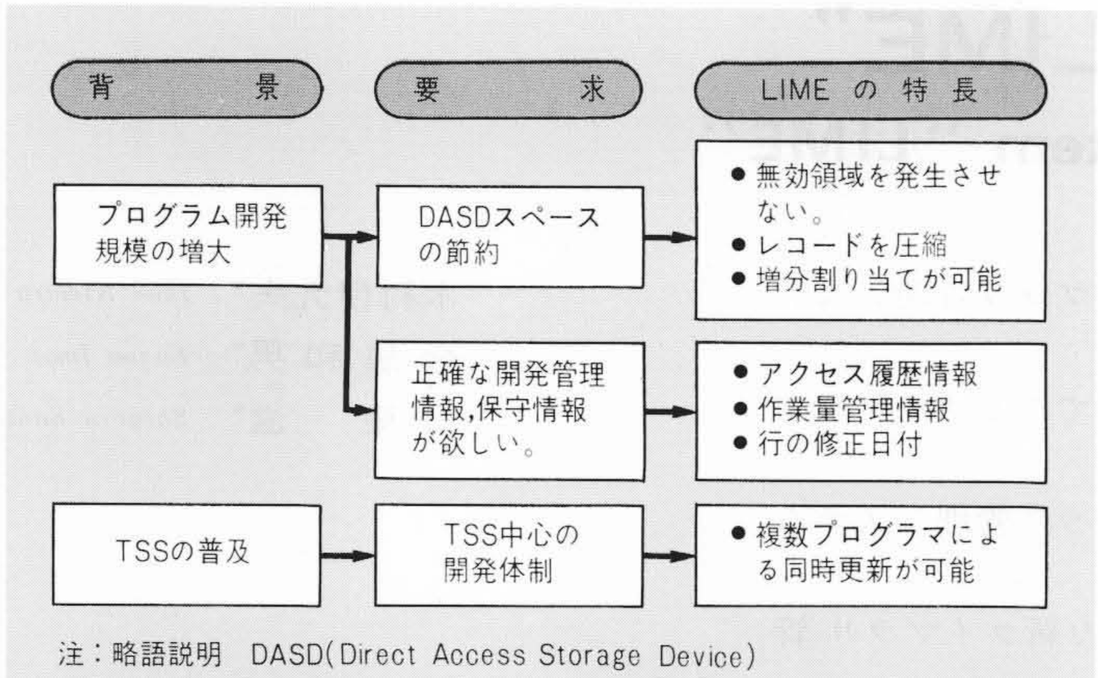

図1 VOS3 LIME開発の背景とLIMEの特長 プログラム開発規模の増大とTSSの普及により，ソースプログラムを効率良く集中管理できるライブラリ管理システムが望まれるようになってきた。

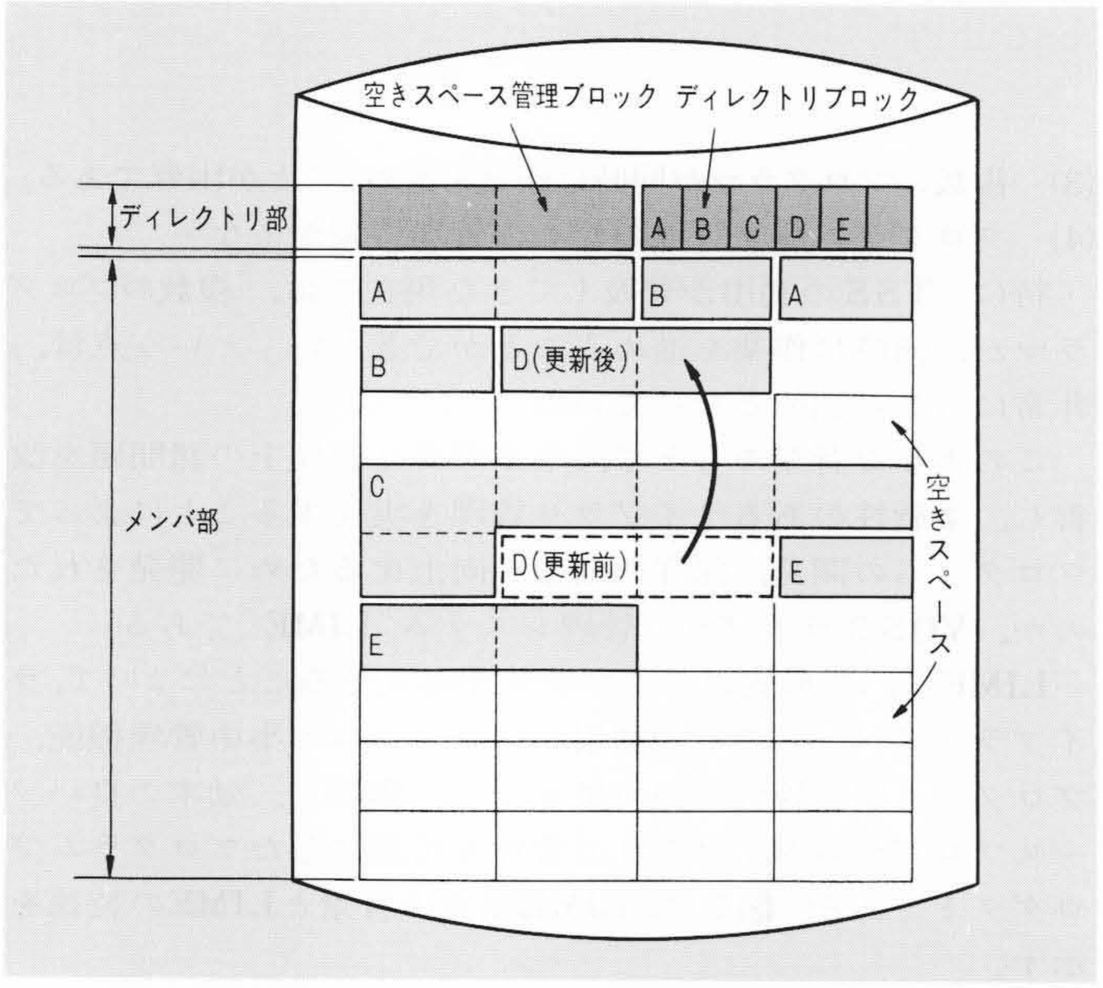

図2 LIMEライブラリの構造 (1)LIMEライブラリの全領域はブロックに分割されている。(2)各メンバは複数のブロックに収められる。メンバを構成するブロックは連続していてもよいし(メンバC)，分散していてもよい(メンバA，B)。(3)ライブラリ中の全ブロックは空きスペース管理ブロックに，空き又は使用中の別が登録されている。(4)各ブロックは使用中から空きの状態に変化すると(メンバDの更新前の領域)，その旨が空きスペース管理ブロックに登録され再利用可能となる。

表1 LIMEにおけるメンバの共用・排他制御の方式 LIMEライブラリのメンバを複数のユーザーが同時に参照・更新する場合にその処理が制限されるのは，他のユーザーが参照・更新中のメンバと同じメンバを更新しようとしたときだけである。したがって，同一ライブラリ上に多数のメンバを作成しても円滑に運用できる。

後続処理＼先行処理		参照	更新
別のメンバ	参照	○	○
	更新	○	○
同一のメンバ	参照	○	○
	更新	△	×

注：○(同時処理が可能である。)
△(先行する参照処理が終了するまで，後続の更新が待たされる。)
×(後続の更新処理がエラーとなる。)

じても，メンバを各断片に分割して格納できる。これにより，ライブラリスペースを完全に無駄なく使い尽くすことができる。

(3) データを圧縮して格納する。

メンバの内容を圧縮して，ライブラリ上に格納する。圧縮の方法には，以下に述べるような手法を用いている。

(a) 連続した空白を圧縮する。

(b) 1文字(8ビット)を6ビットに圧縮する。

(c) COBOLの予約語を圧縮する。

(4) ライブラリの増分割当てが可能である。

ライブラリ作成時には，ライブラリスペースを小さ目に確保しておき，確保済みのスペースが不足した時点で，ライブラリスペースを追加することができる。

3.2 プログラムの集中管理

ソースプログラムは，プロジェクト単位などに一括して集中管理できることが望ましい。しかし，多くのプログラムを一つのライブラリに格納すると，複数のプログラマが同時に処理することができないこと，特定のグループに固有な処理を実行するときにグループ内のプログラムを選択することが難しくなること，などの問題が発生しやすい。

LIMEでは，プログラムを集中的に管理し，かつ個々のプログラマのライブラリアクセスや特定グループに属するプログラムの選択を容易にするために，以下に述べるような機能がある。

(1) 独自の空きスペース管理方式により，複数メンバの同時出力を可能にした。これにより，複数のプログラマが同時に更新作業を実行しても，一方が処理を待たされるなどの不都合は発生しない。

(2) ライブラリアクセスに対する共用・排他制御をメンバレベルできめ細かく実行している。これにより，異なるメンバが同時に参照・更新できるのはもちろんのこと，同一メンバを同時に更新するなどの無意味な作業を事前に検出し，禁止することもできる。**表1**にLIMEでのメンバの共用・排他制御の方式を示す。

(3) ユーティリティにグループ処理機能をもたせている。これにより，ライブラリ内のメンバを特定の属性によりグループ分けし，目的に従って処理の対象を限定したり，拡大したりすることができる。

3.3 プログラム開発管理支援機能

プログラムの開発状況を的確に管理するには，各プログラムの状態を定量的に把握することが重要である。管理の基礎データとなる情報は，一般に，各プログラマが管理者に報告するという形態が取られる。従来のシステムでは，管理者に報告される数値が正確性に欠けたり，プログラマが管理情報を算出するための時間がかかり過ぎる，などの問題があった。

LIMEでは，プログラム開発管理の基礎データとなる情報をライブラリの内部にもたせ，自動的に管理情報を収集する。これにより，プログラムの管理を効率良く，かつ的確に進めることができる。プログラム開発管理支援情報には，以下に述べるようなものがある。

(1) プログラムの参照・更新の回数と日時

プログラムのアクセス状況を把握することができる。

(2) プログラムに対する作業量

作成時の行数，現在の行数，更新した行数，追加した行数及び削除した行数を自動的に記録する。これにより，作業量及び作業の性格を把握することができる。

(3) 修正行の明示

更新された行には，更新日付けを記録する。これにより，

行ごとに更新時期を明確にすることができる。

(4) プログラム管理情報

プログラムの種類，担当者，プログラムの状態などを，プログラムごとに記録しておくことができる。

3.4 メンバの世代管理

一般にプログラムは，完成後も機能追加や仕様変更などの改訂が行なわれるのが普通である。

プログラムを改訂する場合，改訂前のプログラムを保存した上で改訂作業に入る。このため，従来のライブラリシステムでは改訂作業用の別ライブラリを割り当てるなどの工夫が必要であった。

LIMEでは，メンバの世代管理機能により，改訂前後のメンバの内容を同一ライブラリ内に同時に保存することができる。これにより，改訂作業の開始が容易になると同時に，一連の改訂作業の履歴管理も容易になる。

メンバの世代管理の方式を**図3**に示す。

3.5 容易な移行

新しいライブラリ管理システムを導入する場合，従来の管理形態から新しい管理形態に移行するための種々の問題が発生する。区分データセットからLIMEへの移行は，以下に述べるように極めて容易である。

(1) ライブラリの移行

LIMEユーティリティの移行機能により，容易に移行できる。

(2) コンパイルなどのジョブ制御文の移行

ジョブ制御文の形態や手順は基本的に従来と同じであり，変更は不要である。処理するライブラリがLIMEであるかどうかはシステムが判断する。

(3) プログラマの再教育

ソースプログラムの編集に使用するエディタの文法は，従来のエディタと同一であり，再教育は不要である。

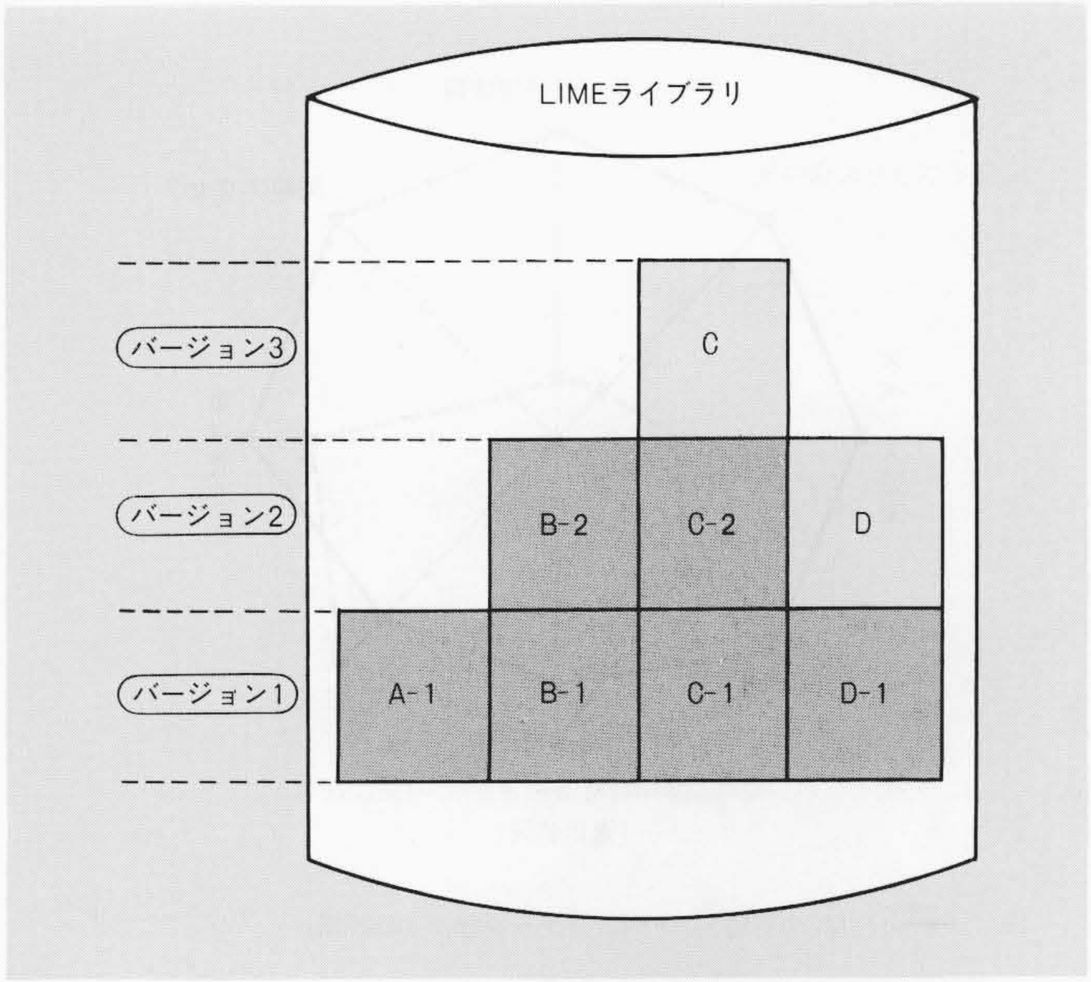

図3 メンバの世代管理の方式(メンバA，B，C，Dをバージョン1から3までの3世代にわたって管理する例) (バージョン1)：メンバA，B，C，Dを作成し凍結する。(バージョン2)：メンバB，Cを改訂し凍結する。(バージョン3)：メンバC，Dを改訂中である。C，Dは改訂作業進行中であり未凍結である。なおメンバDは，(バージョン2)で改訂していないため，(バージョン2)の世代はない。

4 LIMEの構成

LIMEのプログラム構成を**図4**に示す。同図中で■部のプログラムがLIMEとして提供されるものである。また，□部のプログラムは，LIMEとは別に提供されるが，LIMEと関連をもつプログラムである。

LIMEを構成するプログラムと役割について以下に概説する。

(1) LIMEユーティリティ

LIMEライブラリの作成・消去やメンバの複写・バックアップなど，ライブラリ管理用の機能を提供する。

(2) LIMEエディタ

ソースプログラムを編集する。TSS用のエディタとバッチ用のエディタの2種類がある。LIMEライブラリだけでなく，区分データセット，順データセットを編集することもできる。

(3) 構造化プログラミング用画面エディタ“DESP”

“DESP”(Display Editor for Structured Programming)は，ソースプログラムをビデオ端末を用いて編集するためのプログラムプロダクトである。ビデオ端末の特性を生かした効率の良い編集を実行することができる。LIMEライブラリだけでなく，区分データセット，順データセットを編集することもできる。

(4) LIME専用アクセス法

LIMEライブラリの入出力を行なうための専用アクセス法マクロ群である。

(5) 順・区分アクセス法インタフェース

従来の順・区分アクセス法により，そのままLIMEライブラリをアクセスするための拡張機能である。

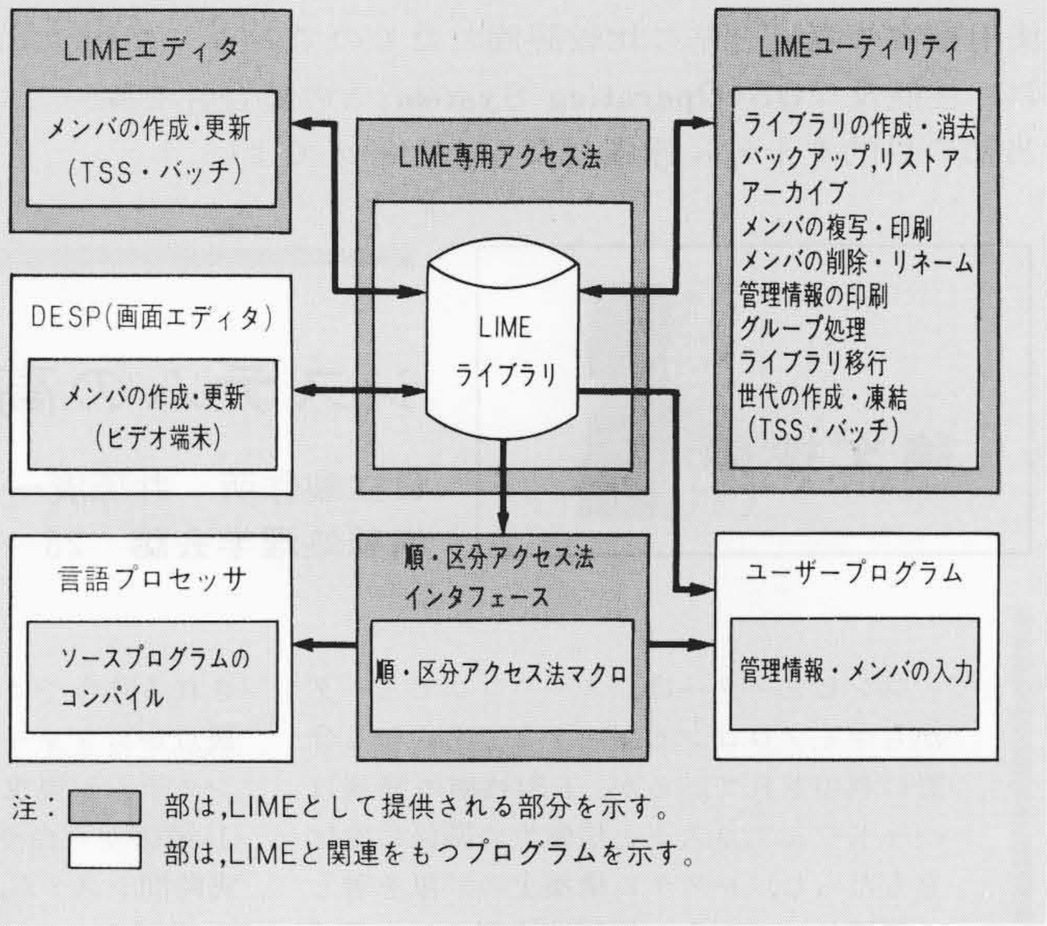

図4 LIMEのプログラム構成と主な機能 LIME及びLIMEに関連するプログラムの構成を示す。

新形式のライブラリを既存のアクセス法で処理できるようにしたものであり，LIMEの大きな特徴の一つである。

(6) 言語プロセッサ

言語プロセッサは，LIMEライブラリを直接入力することができる。

(7) ユーザープログラム

順・区分アクセス法により，区分データセットを参照しているユーザープログラムは，プログラムを変更することなく，LIMEライブラリからデータを入力することができる。

また，LIME専用アクセス法を用いて，LIMEのディレクトリを入力するプログラムを作成することもできる。

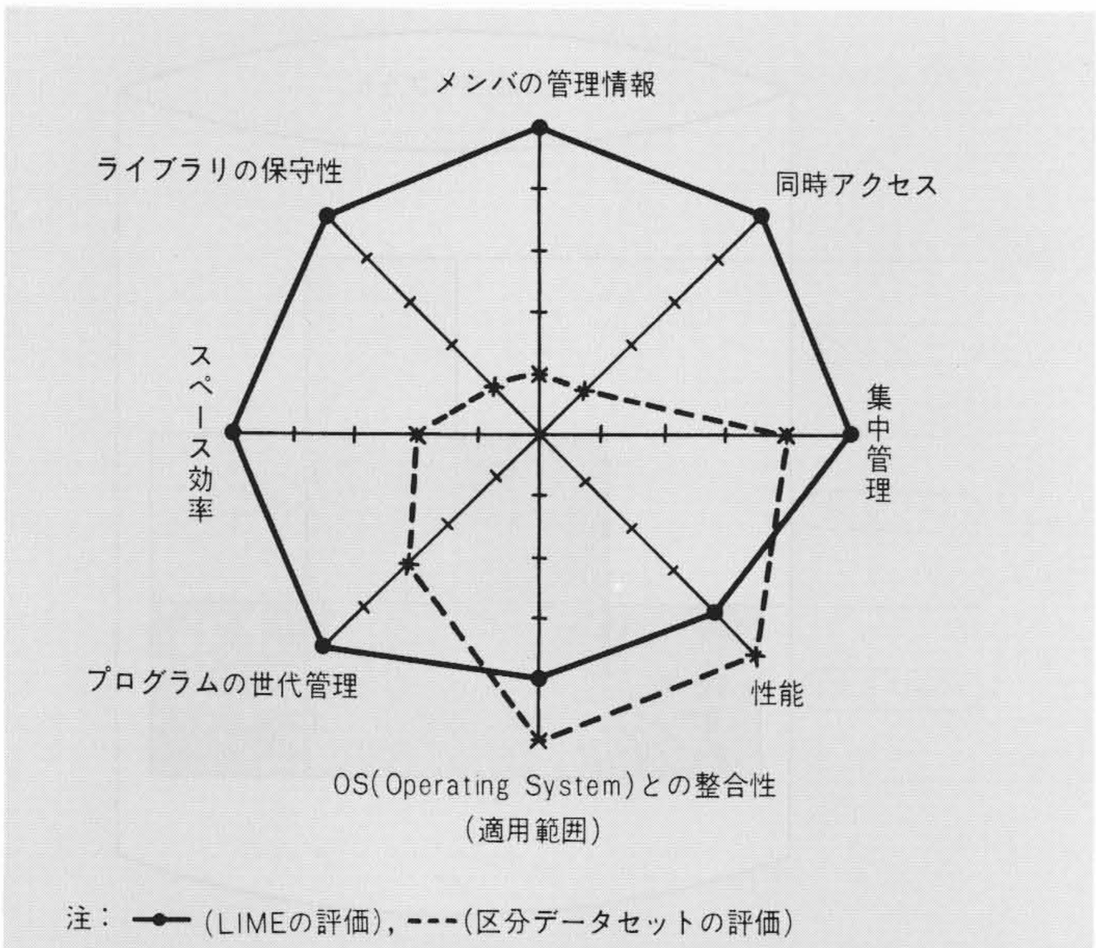

図5 ライブラリ管理システムの評価 LIME及び区分データセットについて，ライブラリ管理システムとしての総合的な評価を示す。

5 LIMEの評価

図5は，ライブラリ管理システムとしての機能を，従来の区分データセットの場合とLIMEを利用した場合とを，LIME使用顧客の意見を基に比較評価したものである。これによれば，性能及びOS(Operating System)との整合性を除いては，当初の目的を十分に達成したものと考えてよい。

性能上の問題については，LIMEでは従来の区分データセットに比べ，ライブラリ構造が複雑化していることが性能低下の要因となっているものである。また，OSとの整合性の悪さの例としては，LIMEライブラリ中に格納可能なデータをソースプログラムに限定していることにより，LIMEの適用範囲を狭めている点を挙げることができる。

6 結 言

本論文では，VOS3ライブラリ管理システム“LIME”の特長と構成について概説した。ライブラリ管理システムの機能は，プログラム開発作業の底辺を支えるものであり，その良否は，プログラムの生産性を左右する重要な要因となる。

LIMEにより実現したプログラムの集中管理機能，開発管理情報の拡充，ライブラリスペースの有効利用などの機能は，ライブラリ管理での基本的な諸問題を解決したものであり，プログラム開発・保守を効果的に行なうために大きな効果を発揮するものである。

LIMEは，昭和56年10月出荷以来，多数のユーザーに利用され好評を得ている。

今後は，更に操作性の改善，機能の拡充，性能の向上などに努力を続け，より良いシステムに発展させてゆかなければならない。また，LIMEの開発により蓄積した技術をもとに，総合的なライブラリ管理システムを実現するために，管理の対象をソースプログラムに限定することなく，広範な一般データに拡張するための研究・開発を重ねてゆく考えである。

参考文献

1) 青山，外：ソフトウェア生産の諸問題とその生産技術の動向，日立評論，62，12，847～856（昭55-12）

論文抄録

システムの高信頼化技術

日立製作所 井原廣一

情報処理学会誌 23—4，327～334（昭57-4）

コンピューターは，スーパコンピュータからマイクロコンピュータまであらゆる分野に利用されているが，LSI技術の発展はハードウェアコストと信頼性の関係に変化をもたらし，システム構築上の制限を著しく軽減しつつある。信頼性技術では，素子の高信頼化からシステム的観点の高信頼化技術が主流になりつつある。一つの方向は，汎用コンピュータでのRAS技術であり，誤りの検知，訂正，再試行，誤り記録などの機能を付加するものであり，他は独立した複数の自律的なサブシステムの結合によってシステムを構築して，フォールトトレラント性をもたせる方法である。超高信頼システムとして故障率が10^{-10}/時間以下のものが現在米国で開発されつつある。

初期の集中システムより，信頼性，処理性の向上を目的にした階層システム，機能分散システムなどの分散システムが出現したが，より実時間で大規模なシステムが導入される昨今では，ライフサイクルコストの観点からフォールトトレラント性をもったシステムが要求されつつある。すなわち，(1)障害が人命や高価な装置に損害を与える実時間システム，(2)保守不可能なシステム，(3)共通データの破壊が絶対許されないシステム，(4)信号処理などの超高性能計算システム，(5)保守コストの高いシステム，(6)長大な連続ライン制御システム，はいずれも超高信頼化が重要である。

コンピュータの開発の初期から信頼性向上問題がつきまとっていたが，比較的コスト制限の少ない軍用や宇宙関係で高信頼システムが開発されている。SAPO, SAGE, RTCCがその代表である。民生用としては電子交換機のESSがある。搭載用としてはOAO計画のPPDS，サターンロケット制御用などがある。深宇宙船用としてはJPL-STARがある。その後ミニ，マイクロコンピュータの出現により，多数のプロセッサ結合方式が開発され，コンピュータネットワークIMPが実用化されている。我が国では新幹線COMTRACがその代表例である。

米国NASA主導の高効率航空機制御用コンピュータは，1977年から開発されている。SRI社の担当するSIFTと，CSDLのFTMPがそれである。SIFTはソフトウェアによる多数決論理方式であり，FTMPはハードウェアによる多数決論理を採用している。いずれも一般のマイクロコンピュータチップを利用しているが，前者はシステムレベル，後者はチップレベルでの多数決方式である。より汎用向けとして開発中のものにUCLAのFTBBCがある。4種の自律的チップにより汎用のマイクロコンピュータ二重系と冗長メモリを結合して，一つのサブシステムを構成する。サブシステム間結合は三重系バスで他へのメモリアクセスも可能であり，サブシステム内も4種のチップにより機能の縮退が自動的に行なわれる故障に強いものとなっている。