プログラム・ドキュメント情報蓄積・管理支援の一方式

落水 浩一郎　森田 亮*　内山 幹康
（静岡大学 工学部 情報工学科）

1. はじめに

ソフトウェア・ライフサイクルにおけるプログラミング・チームの活動は，各種文書情報の生成，変換，蓄積，検索のプロセスとして統一的に把握できる。

我々は，ソフトウェア・ライフサイクルを定義，設計，製造，保守の4局面に大別し，各局面に応じた情報表現の基本単位，および基本単位間の関係を定め，
(1) 計算機2次記憶上にドキュメント情報を蓄積・管理するためのメカニズム，
(2) 蓄積すべき情報をライフサイクルの局面進行と同期させて収集するメカニズム，
(3) 蓄積情報に対して，プログラミング・チームのメンバがアクセスして，有益な情報を効率よく入手するための検索メカニズム
等の検討をおこないつつ，あわせて，
(4) プログラマの思考活動を積極的に支援するタイプのソフトウェア・ツール
(5) 思考活動の阻害要因を除去するタイプのソフトウェア・ツール
を開発，配置するという接近法でプログラミング支援システムを開発中である。

本報告は，主に(1)に関する考察をまとめたものであり，(2)(3)(4)(5)に関しては，製造レベルを対象としたサブシステムを開発済であり別に報告する。[1]

2. 支援対象のプログラム，プログラミング活動および計算機環境

上記のような研究においては，その発想，工夫が，現在利用できる計算機環境，その計算機上で設計実現したプログラム群および日常のプログラミング活動等に基づく経験におおかれ少なかれ影響をうける。本節では，我々の有する上記環境について紹介する。逆にいえば，本節で述べる環境下における実証的研究の下で，ソフトウェア工学上のより一般的問題を解決していくのが我々の意図である。表1に，我々の計算機環境を示す。**

計算機	主記憶	補助記憶	利用端末他	オペレーティング・システム	主力言語	主な開発プログラム
FACOM 230-45S	224KB	Disk 100MB MT 1台	TTY CRT LP PDP 11/45	OSII VS (バッチ・システム 小規模オンライン可能 の構成)	PL/I アセンブリ言語	コンパイラ，オートフローチャータ[2]，ファイルユーティリティ MTO シミュレータ，オンラインテキストエディタ ユーザビリティチェッカ & アナライザ[3] プログラム蓄積・管理支援システム[4] アルゴリズムレベルのプログラム文書自動化ツール[1]
MELCOM 70	32KW	Disk 10MB	TTY CRT 通信制御装置 NCU, MODEM	R-DOS (リアルタイム・オペレーティング・システム)	アセンブリ言語	コンパイラ，インタプリタ，逆アセンブラ，エディタ (現在，コンピュータネットワーク "T-Net"[5] をインプリメント中)
PDP 11/45	8KW	Disk 2.4MB	DEC writer CRT	DOS	アセンブリ言語	I/O ハンドラ，カセットファイルシステム
LSI 11	8KW	カセットMT	CRT	なし	アセンブリ言語	

表1 計算機環境

表2に，開発プロジェクトチームに関する環境を示す。

表1，表2に示す環境は，一般に，大学の研究室等の環境に相等すると思われる。表3に，我々が現在採用しているソフトウェア作成の手順，採用している方法論，採用予定の方法論，利用しているソフトウェア・ツール，開発予定のソ

* 現在，三菱電機株式会社勤務
** 我々の研究室に関係するもののみを記入してある

開発対象プログラムの規模	$\alpha \times 10^4$ ステップ	$0.3 < \alpha < 5$
チーム構成員数 チーム数	4～5名 2～3チーム	ただし，総人数は 10～12名
開発期間	1～2年・チーム	ただし，複数年にわたるときは，チーム構成員の半数以上が入れかわり，新人教育に 2～3カ月を要する。

表2　プロジェクト・チームに関する環境

ソフトウェア・ツールの大要を示す。

ステップ	作成するドキュメント類	目標	方法論	(ソフトウェア)ツール
1 (定義)	利用手引書	作成すべきソフトウェアの外部的要件を，利用者の立場にたって十分に議論する	なし	紙と鉛筆，黒板 複写器
2 (定義)	システム設計書 ① データフローと機能の関連図 ② 機能の制御的関連図 ③ スケジュール表	① 主要データ，データ流と機能の関連の定義 ② 機能(プロセス)間の制御の定義	① なし (ISDOS) ② なし (SREM[6])	同上 (定義レベル生成器 〃　提示器)
3 (設計)	プログラム設計書 {モジュール設計書 インタフェース設計書 ファイル設計書 データ設計書 モジュール外部仕様書	独立性の高いモジュール構造の実現 (独立性) 作成時…インタラクションの少ない作業分割 検査・保守時…誤りの局所化と波及効果の減少	複合設計[7] エゴレス・プログラミング[9]	同上 (定義レベル提示器 設計レベル生成器 設計レベル提示器)
4 (製造)	モジュール内部仕様書 ソースリスト ジョブ制御情報 ファイル情報	整構造プログラムの作成	段階的詳細化技法[8] エゴレス・プログラミング	オペレーティングシステム オンラインテキスト・エディタ 製造レベル生成器[1] 製造レベル提示器 (設計レベル提示器)
5 (保守)	障害レポート 修正レポート	虫の早期発見と回復	なし	なし (コンピュータ・メイルボックス 各レベル提示器 検査システム)

表3　現在および検討・作成中のソフトウェア開発手順，(　)内は検討中

3、ドキュメント情報蓄積によるプログラミング支援

表3に示す局面にそってプログラム開発を進める場合に生じる問題点は以下の通りである。

(1) ステップ1～5を通じて，ドキュメント(設計活動の結果，得られる情報)は主に紙を媒体にして人間の手作業により作成される。この方式では，表現の柔軟性，および必要に応じて完全性の高いドキュメントを作成できる利点がある。反面，加筆修正，さしかえ等の更新作業が容易でない。また，必要な情報を検索する労力が多大になる。検索を容易にするため，索引を十分に設けたり，情報内容を重複してあちこちに記入すると，さらに更新時の労力を増す。

(2) ステップ1～3においては，誤りの訂正や，より適切な代替案の発生等にともなう修正作業（局面内での修正フィードバック・ループと呼ぶ）がとくにひんぱんである。　十分な議論をへて，例えば，システム設計書が完成し清書したあとで，さらにより適切な代替案が提案され，それをしりぞける結果になることすらある。　また，細かな誤り等の修正は，清書原稿に加筆修正したままで再清書をしないことが多い。

(3) Top-down 開発の欠点として，下位レベルの局面での検討が，上位の局面における決定の大変更や小変更をもたらすことがある（局面間での修正フィードバックループと呼ぶ）。　この時，その局面以前の清書ドキュメントが大中に変更をうけることになり多大の労力を有する。

(4) とくに保守時の情報検索の特徴として，多局面にわたる情報の従のつながりを bottom up に把握する必要が生じる（例えば，運用中のプログラムに障害が発生し，原因があるモジュールに特定されたとき，誤りを検出し，波及効果も含めて正しく訂正する場合で作成者と保守者が異なるとき）。　一般に，ドキュメント類は Top down 開発時の作業進行にそってまとめられており，それにそぐわない検索パターンに対しては検索効率が極端に悪化する。

第1節で述べた考え方と上記の問題解決をはかるため，図1に示すシステムを現在構築中であり，一部は完成している[1]。

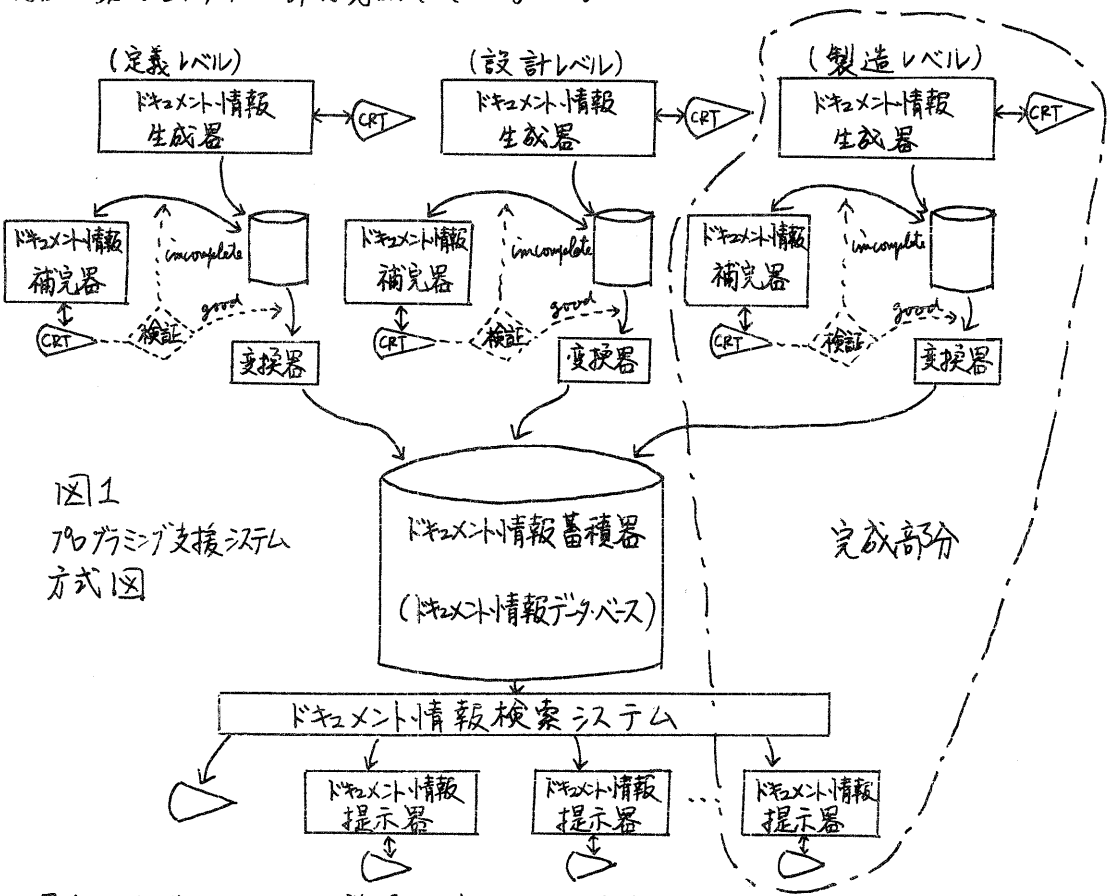


図1
プログラミング支援システム方式図

図1のサブシステムの説明を以下におこなう。

(1) 情報生成器　開発チームのメンバにより，生成される設計方針，実現方式等に関する情報を，生成者本来の活動（プログラミング）を支援しながら収集する。　定義，設計，製造の各局面毎に情報形体が異なるため，生成器は各々用意する。　情報表現の手段としては，情報記述を主目的とする計算機言語に依る。
(2) 情報提示器　蓄積情報を利用者の要求に応じた表現形式で会話的に提示する。　利用者（作成，検査，保守）毎に必要とする情報集合が異なるので(イ)利用者の代表的検索パターンをあらかじめ想定した提示システム，(ロ)利用者の要求毎に直接検索提示とおこなう部分に分ける。
(3) 情報補完器　利用者が満足する情報を提示するには，生成器によって得られる情報の質と完備性を高める必要がある。　補完器は，蓄積器が要求する情報の補足，見直しによる高品質化を情報生成者に要求する。
(4) 情報蓄積器　提示器の利用者は，その目的毎に，生成器の生みだす情報集合とは異なった分類の情報集合を要求する。　この間のギャップをなくし，3節(1)～(4)で述べた，ドキュメント情報管理上の問題点を解決するためデータ・ベースとして実現する。　蓄積器については次節以降に詳述する。

4. ドキュメント情報蓄積器

図1のシステムの中心的存在は，ドキュメント情報を蓄積・管理するメカニズム（ドキュメント情報蓄積器）である。　蓄積器は，効率的情報の収集，蓄積・管理，提示がおこなえるように，以下に示す特徴を有するように設計されねばならない。

（1）種々の報告書作成が容易におこなえる。
（2）蓄積情報の変更作業が，実際のソフトウェア開発・保守作業に十分追従できる。
（3）情報内容が十分である。

このような要件をみたす蓄積メカニズムの設計について，4－1で述べる。
また，ドキュメント情報を管理する上での固有の問題点について4－2で述べる。

4.1 蓄積情報の構造

情報蓄積器はデータ・ベースとして実現する，データ・モデルとしてはE-R（entity-relational）モデル[10]を採用する。　表4にentity setを示す。　図2にRelationship setsを示す。　表4，図2に示したentity set，relationship setおよびそれらのattributeは表3をもとにして，ごく基本的（必要最小限）なものを洗いだした。　実際のドキュメント情報を蓄積利用する過程でより精選・追加していく予定である。

4.2 ドキュメント情報管理における諸問題

ドキュメント情報を管理する際，次に示す問題点が存在する。

(1) ドキュメント情報の変更管理
(2) ドキュメント情報のバージョン管理
(3) ドキュメント情報生成・利用者のアクセス制御

すなわち，ソフトウェアの開発はライフサイクルの各局面をへて進行するが，

分類	entity set	predicate	attribute
プロジェクト	Project	同一研究のために集まった人達の集合	プロジェクト名，目的，スケジュール
	Person	データベース利用者	姓名
プログラム	system	同一の研究のために開発されるプログラム群	システム名，機能，計算機環境，実行情報，利用情報
	program	メイン・プログラムを中心とした プログラム・モジュール，ファイルの全体	プログラム名，機能，総合編集情報
	module	1コンパイル単位	モジュール名，機能，言語，コンパイル情報，ソースリスト
ステートメント	block	一連のステートメント(構文単位)	範囲，レベル番号，機能
	statement	1ステートメント	文番号，文の種類，機能
データ	unstructured	主記憶上の領域で構造をもたないもの	データ名，型，大きさ，初期値，値の範囲
	structured	主記憶上の領域で構造をもつもの	構造名，大きさ，アクセス法
	file	補助記憶上の領域	ファイル名，格納媒体名，ファイル形式，レコード形式

表4　entity sets

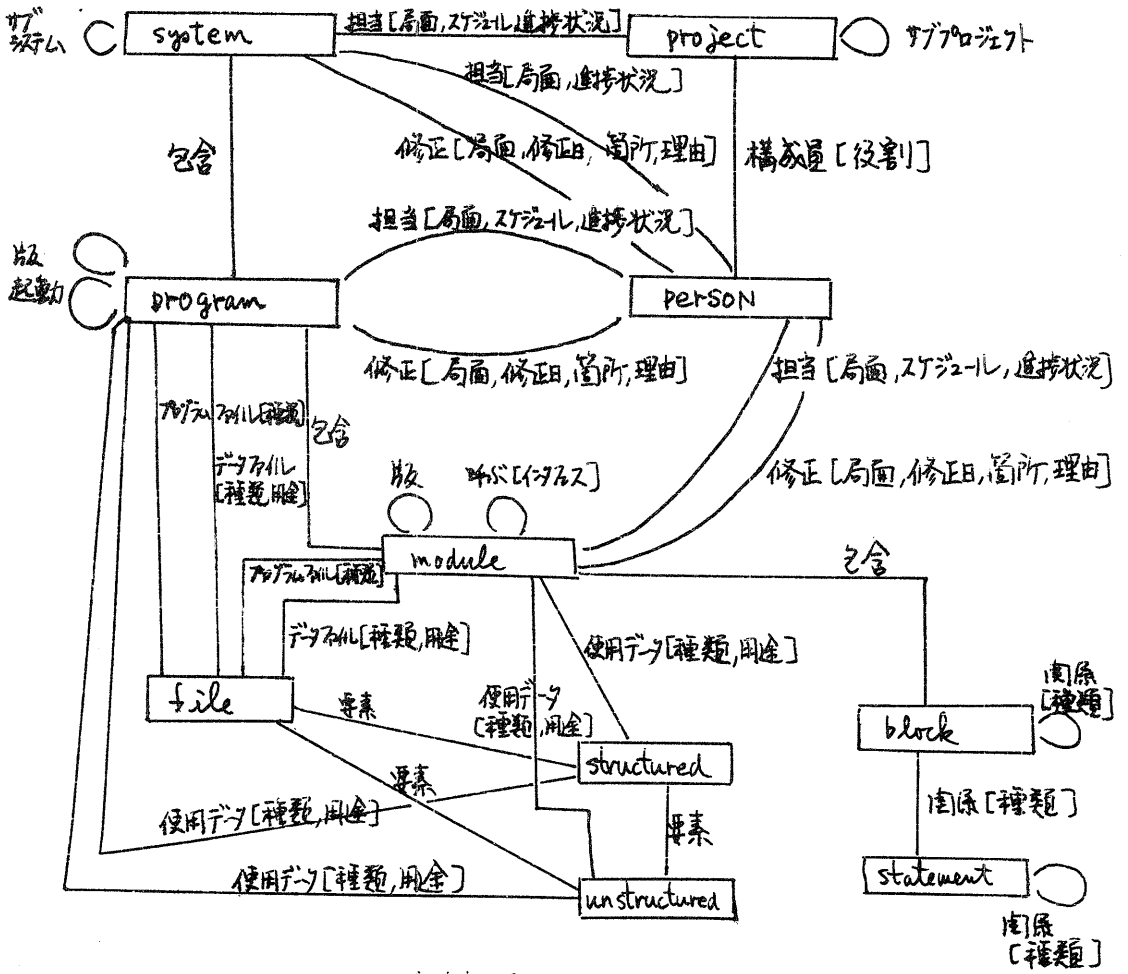


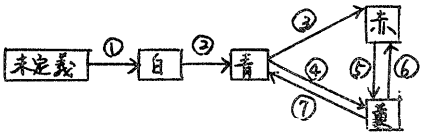
図2　relationship sets

通常，誤りや，より適切な代替案の発生，要求の変化等による変更のため，局面内や局面間にわたるフィードバックループを形成する。このため蓄積器内に常に矛盾したデータが含まれる状態が多発し，それに対して，利用者がアクセスして混乱を招く危険性が生じる。これを防ぐためには，変更要求が起った場合，内容が変更される情報単位に対する利用者のアクセスを制御する必要がある。また，通常のソフトウェアに対しては，複数バージョンのソフトウェアが生産され，それらを管理する必要性も高い。

これらの問題を解決するために，図2において，system，program，module，fileのentity setの各entityに状態情報を付加する。図3に各entityが取る状態とその意味，および状態遷移図を示す。

状態	意味
赤	修正中あるいは修正必須のentity
黄	修正の可能性のあるentity
青	定常状態のentity
白	作成中のentity
未定義	未定義のentity

(a) 状態の定義



①	entityの作成が開始される時
②	entityの作成が完了した時
③	entityに対する修正要求が承認された時
④	「赤」のentityと図2で示されるrelationshipで直接関連づけられたとき
⑤	entityの修正作業が終了した時
⑥	「赤」のentityの修正による波及をうけて，修正の必要が生じたとき
⑦	すべてのentityの修正作業が終了した時

(b) 状態遷移

図3 状態の定義と推移

4.2.1 ドキュメント情報の変更管理

4.2.1.1 局面内での修正フィードバックループの管理

ソフトウェア・ライフサイクルにおける1つの局面では，それ以前の局面で生成された情報を利用して，それを詳細化，具体化する作業がおこなわれる。そこでは，最適解を見つけるために様々の提案，代替案が提出され，まさに時々刻々と情報が変化している。そのような状況下では，蓄積情報の生成者と利用者との間で，蓄積器を経由しない情報のやりとりがおこなわれる危険性が生じる(図4)。しかも，情報の生成者や利用者は，その方が便利であると感じるに違いない。

そこで，この問題を解決するために，entityの状態に「白」の状態を導入した。すなわち，「白」の状態にあるentityは作成中であることのみを蓄積器は管理し，作成内容は管理外である。それに対するアクセスも情報生成者と情報利用者に委ねられる。

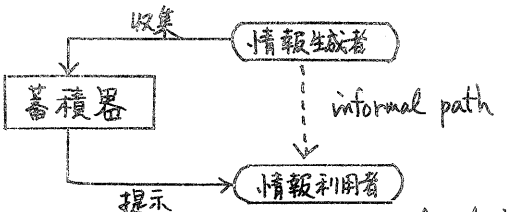


図4 蓄積器を経由しないinformal pathの出現

4.2.1.2 局面間での修正フィードバックループの管理

局面間での修正フィードバックループによる，情報管理の方針を，図5を利用して説明する。今，モジュールentity M_1，M_2 が「白」の状態で，残りのentity

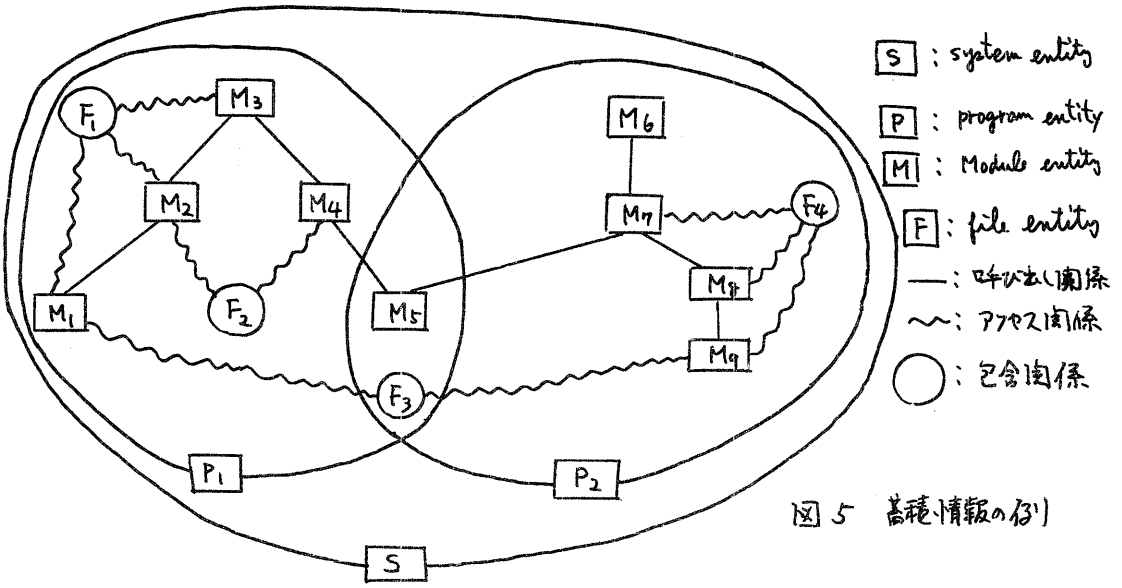


図5 蓄積情報の絵

は，すべて「青」の状態にあるとする。 ここで，プログラム entity P_1 に変更要求が生じ承認されたとする。 ここで，P_1 の状態は「赤」になる。 すると，P_1 と直接関連づけられている，システム entity S，モジュール entity M_3, M_4, M_5（「白」の状態にある M_1, M_2 は除外される），ファイル entity F_1, F_2, F_3 の状態が「黄」になる。 この状態で，P_1 の修正作業が進められ，4.2.3 節で述べる基準に従ってデータベース利用者の一部はアクセスに制限をつけられる。 この修正の波及のために，たとえば F_1 の修正が必要になった場合，その状態は「赤」になる。この場合も前と同様に，F_1 と直接関連づけられていて，かつ「青」の状態にある entity は「黄」の状態になる。 この場合，M_3 はすでに「黄」の状態になっている。 この状態で，F_1 の修正作業が開始される。 このようにして，すべての修正作業が完了し，それ以上の波及が生じなければ「黄」の状態にある entity はすべて「青」の状態に戻される。

4.2.2 バージョン管理

通常，機能改良，性能改善，操作性の向上等の要求発生により，モジュールレベル，プログラム・レベルで，バージョン・アップの要求が出現する。 これに対処するため，module entity, Program entity 間に「版」という relationship を設定する。

しかし，新バージョンへの切り替えのタイミング，旧バージョンの処置等について，データ・ベース内情報の量的発散を押える意味から検討する必要がある。

今後の検討課題とする。

4.2.3 情報生成・利用者のアクセス制御

蓄積情報に対するアクセスは次の4つのパラメータにより定まる。

(1) アクセスされる entity set, relationship set

(2) アクセスされる entity, relationship の状態（ただし，本報告では entity の状態のみを考慮している）

(3) アクセスされる entity, relationship に対するオペレーション

（4）アクセスを行なう情報生成・利用者の役割（定義者，設計者，プログラマ，検査者，ユーザ，保守者）

ここでは，entityに対するアクセス制御の一方式を表5に示す。また，entityの状態変化を要求できる情報生成者，利用者について表6にまとめる。要求の承認はシステムがチェックしたのちになされる。

利用者 ＼ entity 状態 ＼ オペレーション		system entity				program entity				module entity				file entity			
		赤	黄	青	白	赤	黄	青	白	赤	黄	青	白	赤	黄	青	白
定義者	R	◎	◎	◎	◎	×	×	×	×	×	×	×	×	×	×	×	×
	W	◎	×	×	◎	×	×	×	×	×	×	×	×	×	×	×	×
設計者	R	×	○	○	×	◎	◎	◎	◎	○	○	○	○	◎	◎	◎	◎
	W	×	×	×	×	◎	×	×	◎	○	×	×	○	◎	×	×	◎
プログラマ	R	×	×	×	×	×	○	○	×	○	○	○	○	×	×	×	×
	W	×	×	×	×	×	×	×	×	○	×	×	○	×	×	×	×
検査者	R	×	○	○	×	×	○	○	×	×	○	○	×	×	×	×	×
	W	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
ユーザ	R	×	×	×	×	×	○	○	×	×	×	×	×	×	×	×	×
	W	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
保守者	R	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
	W	◎	×	×	◎	◎	×	×	◎	◎	×	×	◎	◎	×	×	◎

R；entityが管理する情報の内容変化を伴わないオペレーション
W；〃　　を引き起こすオペレーション
◎；〃　内容すべてについてアクセスが許可される
○；〃　内容の一部　〃
×；〃　内容に対してアクセスが許可されない

表5　データ・ベース利用者のアクセス制御

	system entity	program entity	module entity	file entity
定義者	○	×	×	×
設計者	×	○	○	×
プログラマ	×	×	○	×
検査者	×	×	×	×
ユーザ	×	×	×	×
保守者	○	○	○	○

表6　entityの状態変化を要求できる情報生成・利用者

○；要求できる
×；〃できない

データ・ベース利用者は表4に示すように，ある状態下では，情報へのアクセスが拒否される。この時，拒否の理由を知りたいことがある。アクセスを拒否される場合としては，

(1) 本来，アクセスが許可されていない。

(2) entityの状態変更により一時的にアクセスが許可されない

の2通りが考えられる。とくに，2の場合，修正者が，修正理由を，コンピュータメイルボックスにより，アクセスを拒否された利用者に伝達することで拒否理由を公開し，修正作業後にはメイルボックスの電文を蓄積情報に組み込むような方式を計画している。

5. おわりに

ドキュメント情報を計算機2次記憶上に蓄積・管理することによって，プログラミング・チームの定義・設計，製造の作業を支援する方式を述べた。

2節で述べた理想，種類のソフトウェアを大学研究室レベルの組織で開発管理していく観点からは基本的な問題点の洗いだしと解決への第一歩は得られたものと思う。現在，設計レベルの生成器，補完器，提示器および蓄積器を設計，実現中であり別に報告する。本文中，表4，図2，表5，表6の内容については，実験システムの利用経験を通じて改善していきたい。

6. 文献

[1] 落水，酒井，八木，国本，"アルゴリズム・レベルのプログラム文書自動化ツール"，情報処理学会，ソフトウェア工学研究会資料，1979年7月。

[2] 落水，福村，"構造的プログラミング用フローチャートに基づくフローチャート自動作成の一方式"，昭和50年度情報処理学会全国大会，1975年11月。

[3] 落水，福村，"リーダビリティチェッカ，アナライザ方式によるプログラム文書化"，昭和51年度情報処理学会全国大会，1977年10月。

[4] 落水，森田，"プログラム蓄積・管理支援システム"，昭和52年度情報処理学会全国大会，1977年10月。

[5] "公衆回線による計算機ネットワークの研究"，文部省特定研究「情報網システムに関する基礎的研究」B-14班：計算機ネットワーク研究グループ報告書，1978年

[6] "SREP final Report" Vol.1 ~ Vol.5，U.S. Department of Commerce National Technical Information Service，1977年8月

[7] G.J. Myers 著，久保，国友共訳，"高信頼性ソフトウェア複合設計"，近代科学社，1976年

[8] N. Wirth 著，野下，筧，武市共訳"系統的プログラミング入門" 近代科学社

[9] G.M. Weinberg，" The Psychology of Computer Programming"，VAN NOSTRAND REINHOLD Company，

[10] Peter PIN-SHAN chen，" The Entity-Relationship Model - Toward a Unified View of Data"，ACM TRAS. on Database System，Vol. 1，No. 1 March 1978，pp 9~36