TENSE OMS における細粒度情報モデル

西岡健自† 平田陽一郎† 渡邊多恵子†

オブジェクト管理システム (OMS) は，ソフトウェア開発支援環境 (SEE) を情報の側面から統合する要である．そして，我々の開発した OMS の目標はさらに，設計書やプログラム等のドキュメント相互の情報の整合性を詳細に保証することである．この OMS を TENSE OMS と呼ぶ．TENSE OMS の基盤として，我々はドキュメント間で重複する情報を識別できる細粒度の情報モデルを提案する．この細粒度の情報によって情報の一元管理が可能となり，所期の目標を実現することができた．TENSE OMS の開発支援における特徴は重複する情報に起因する各種の機械的作業の自動化である．ここで，機械的作業とは中規模以上の開発プロジェクトで多く発生する記述の繰り返しや，変更の反映作業等である．運用における特徴は柔軟なカスタマイズ機能と実用上十分な情報アクセスの応答性である．カスタマイズ機能は情報モデルで独自に定義した拡張フレーム表現に基づき，応答性は Prolog を情報モデルの実装言語として採用したことに基づいている．カスタマイズ機能により TENSE OMS は多様なソフトウェアプロセスに容易に組み込むことができ，SEE の進化にも追随できる．本論文では TENSE OMS の情報モデルと開発支援機能について述べる．なお，機能については TENSE OMS のカスタマイズによって実現した SEE に即して述べる．この SEE を統合化Cプログラミングシステムと呼ぶ．

Fine-grained Information Model on TENSE OMS

Kenji Nishioka,† Yoitiro Hirata† and Taeko Watanabe†

We propose a fine-grained information model for software documents. Based on this model we developed an object management system named TENSE OMS to support efficiently software development activities. TENSE OMS guarantees the consistency among documents by an effect of the fine-grained information. And the guarantee realizes to automate manual activities like duplicate describing and maintenance of documents caused by duplicate information among documents. We adopted an extended frame representation model as a base of the information model, so it is easy to customize TENSE OMS for various software processes. And we adopted Prolog as an implementation language for TENSE OMS, so the data access performance is enough for practical use. In this paper we describe the information model, and various productivity improvement functions of TENSE OMS through Integrated C Programming System, which is a software engineering environment based on customized TENSE OMS.

1. はじめに

ソフトウェアの開発は，複雑さへの挑戦という側面をもちながら，人手に頼る部分が大きい．したがって，複雑なものを人に分かりやすくする手法が発展してきた．その手法の基本として，段階的詳細化やモジュール化[13),14)]が開発現場に定着している．

これらの手法によって，ソフトウェアの構造は単純となり，ドキュメントも読みやすくなる．そして，分かりやすさは，ソフトウェアの信頼性と保守性の高さに通じる．ここで，ドキュメントとは，ソフトウェア開発過程で現れるソースプログラム，設計書，仕様書等をさしている．

しかし，分かりやすさの代償として，ドキュメントには多くの重複する情報が現れる．たとえば，段階的詳細化によると下流の開発フェーズのドキュメントは上流のドキュメントの内容の多くを受け継いでいる．また，モジュール化によると各モジュールは他のモジュールとの相互関係情報を重複して共有する．

この情報の重複が記述の繰り返しや変更の反映作業等の機械的作業を発生させ，知的な作業を阻害する要因となる．さらに多くの場合重複する情報はフェーズの上下やモジュール間で情報の食い違いを招き，信頼性の劣化や手戻りを発生させる[14)]．この情報の重複に伴う弊害は中規模以上のプロジェクトに特に著しい．

この解決法のひとつは上流ドキュメントの情報まで含め，すべてソースプログラムに記述して一元管理す

† 横河電機(株)オープンシステム研究所
Corporate R & D, Open Systems Laboratory,
Yokogawa Electric Corporation

る方法である．しかし，この方法では段階的詳細化を伴わないドキュメント構成になるため第3者による保守が困難となり，モジュール間の情報の食い違いも解決できない．

もうひとつの解決法は上記の機械的作業の自動化である．しかし，機械的作業と詳細化やモジュール化等の知的作業との切り分けは通常困難であり，自動化は遅れている．

そこで，本論文では情報の重複に伴う機械的作業を自動化可能な問題に帰着させる情報モデルを提案する．このモデルは重複した情報を認識し一元化するのに十分な細かい情報粒度を備え，開発途上の未完性のドキュメントを表現する能力をもっている．我々はこの情報モデルに基づいてドキュメント相互のきめ細かい整合性を保証し，上記の機械的作業を自動化するOMS[2]を開発した．このOMSをTENSE☆ OMSと呼ぶ．

TENSE OMSはSEEを情報の面から統合する点では，6章の関連研究で述べるようにPCTE[8]のOMSやCASE[9]のレポジトリ等と共通点をもっている．しかし，TENSE OMSは情報モデルの情報の粒度を上げることによって，ドキュメント間の情報の整合性を詳細に保証する機構を実現している．

運用上の特徴として，TENSE OMSは情報モデルの特性に基づく柔軟なカスタマイズ機能を実現しているため，多様なソフトウェアプロセスに組み込むことができる．また，Prolog[11]を実装言語として採用したことにより，実用上十分な情報アクセスの応答性を達成している．

以下，2章で情報モデルの概要，3章でその情報単位である要素オブジェクトの拡張フレーム表現とPrologによる実装を述べる．また，4章で拡張フレームに基づくカスタマイズ機能の実際，5章でカスタマイズで実現した統合化Cプログラミングシステム[18]の機能と性能評価，6章で関連研究について述べる．

2. 情報モデルの概要

TENSE OMSの情報モデルではソフトウェアを構成する情報単位を要素オブジェクトと名付ける．ドキュメントに現れるシステム，ファイル，モジュール，データ，手続き，引数等の個々の実体，および，実体の階層的な分類における個々の分類情報が要素オブジェクトである．実体に関するものをインスタンス要素オブジェクト，分類に関するものをクラス要素オブジェクトと呼ぶ．以下，両者をそれぞれクラス，インスタンスと略称する．

個々のインスタンスは特定のクラスに属し，ドキュメントに現れるデータタイプやコメント等の属性情報や他のインスタンスとの関係情報を一括して保持する．クラスは書式情報等の下位のクラスやインスタンスで共通な属性情報と，他のクラスとの関係情報を一括して保持する．インスタンスの部分関係から要素オブジェクトは**図1**のように分類できる．

要素オブジェクトは3章で述べるように一種のフレーム[10]である．一般にフレームの情報の最小構成要素をスロットと呼ぶ．個々のスロットには属性情報や関係情報がはいる．したがって，ドキュメントはインスタンスのスロットの内容を書式に沿って配置したものと考えることができる（**図2**）．なお，未完成のドキュメントはそれを構成するインスタンスやスロットの一部が未定義なものとして表すことができる．

また，情報の一元性を実現するために異なる要素オブジェクトでは同一の属性情報を共有しない．したがって，インスタンスはドキュメントを情報の重複の識別可能な細かい粒度で分解し，それらを重複しないように再構成したものと考えることができる（図2）．なお，TENSE OMSでは一個以上のドキュメントでの同一インスタンスの同一属性の複数回の出現を情報の重複と見なす．

要素オブジェクトの相互関係はインスタンスからのドキュメントの復元，情報の変更による影響解析，クラス構成のカスタマイズや保守の効率化等に必要な情報である．本情報モデルではこれらを実現するための最小限の相互関係として部分（BMO），参照（REF），記述（SPC），汎化（AKO）の四つを定義する．

BMOはインスタンス間の階層的な一対多の部分関係である．プロセスとその構成モジュール，手続きと

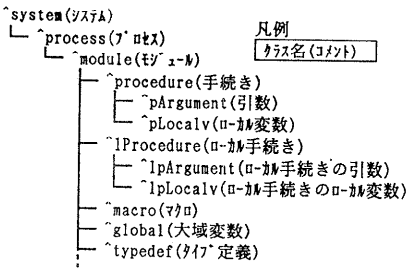

図1 部分関係による要素オブジェクトの分類
Fig. 1 Classification of Elemental Object based on "a part of" relationship.

☆ Total ENvironment of Software Engineering.

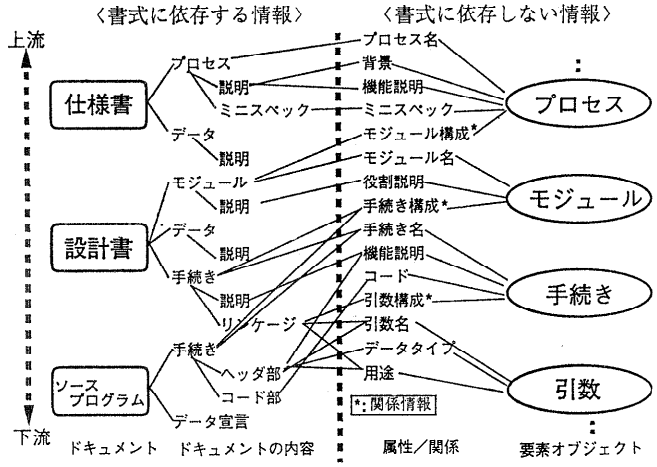

図 2 要素オブジェクトとドキュメントの対応
Fig. 2 Correspondence between Elemental Objects and information on documents.

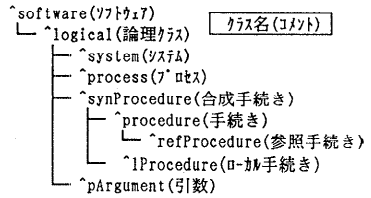

図 3 汎化関係による要素オブジェクトの分類
Fig. 3 Classification of Elemental Object based on "a kind of" relationship.

その引数やローカル変数が BMO の例である．この部分関係によるクラスの分類がさきにあげた図1である．

この図で先頭の "^system" クラスは固定名称のインスタンス，"system" をもち，他のすべてのインスタンスは "system" を頂点とする一個の BMO 関係木を構成する．この木ではソフトウェア開発のフェーズが進むに連れて，頂点から順に葉にあたるインスタンスが明らかになってゆく．また，各スロットの内容は各フェーズで重複する情報を一元化しながら徐々に埋まってゆく．

REF はインスタンス間の多対多の参照関係である．モジュールとそこで外部参照宣言している大域変数，手続きとそこで呼び出す手続きの関係が REF の例である．

SPC はドキュメントファイルのような物理的なインスタンスと，モジュールのような論理的なインスタンスの多対一の記述関係である．プロセス仕様書ファイルとそこに記述してあるプロセスの関係が SPC の例である．

AKO はクラス間，および，クラスとそれに属すインスタンスとの一対多の階層的な継承関係である．この継承関係によるクラスの分類が**図 3** である．

この図で "^refProcedure" は手続きの呼び出し宣言情報を保持する仮想的なクラスである．このクラスのインスタンスは一時的に現れるが，手続きの定義である "procedure" クラスのインスタンスとの照合を受けるのみで消滅する．呼び出す側のインスタンスの REF 関係情報には "procedure" のインスタンス名がはいる．なお，照合で食い違う場合は警告表示が現れ，被呼び出し手続きが未定義の場合は "procedure" のインスタンスを生成する．本情報モデルはこのような仮想的なクラスの導入により同一フェーズで重複する情報を一元化する．

以上四つの相互関係例を**図 4** に示す．

TENSE OMS はすべての要素オブジェクトを TENSE データベース[17]と呼ぶデータベースで管理する．このデータベースを介して TENSE OMS はドキュメントと要素オブジェクトを相互に変換し，ソフトウェア開発過程をとおして常に両者を一致させるように管理する．要素オブジェクトの情報の一元性により，このような相互変換機構がドキュメント相互の整合性を保証する．

この機構ではドキュメントと TENSE データベースとのインタフェースをフィルタと呼ぶ．フィルタはドキュメントを要素オブジェクトに分解する抽出，それに

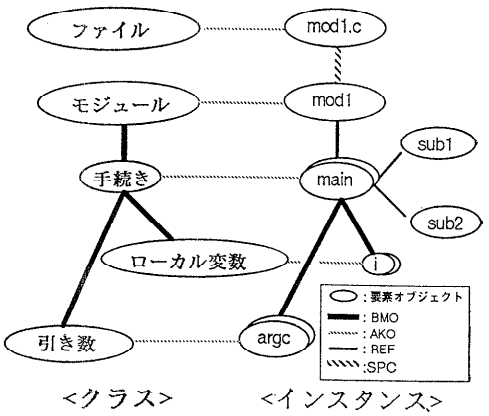

図 4 要素オブジェクトの相互関係
Fig. 4 Relationships among Elemental Objects.

基づいてデータベースを更新する格納，データベースの内容をドキュメントに変換する逆生成[19]からなる．

格納処理はドキュメント変更に伴う影響解析や，ドキュメントファイルへの影響通知も行う．影響解析は要素オブジェクトの四種の相互関係に基づく探索の形をとる[17]．影響通知を受けたドキュメントを逆生成すると情報の一元性により変更の自動反映が実現する．

この逆生成処理により，未記述のドキュメントを既存の要素オブジェクトから部分的に生成することができる．新たな書式を定義すれば既存の要素オブジェクトから新しいドキュメントも自動的に生成できる．

以上のような整合性の保証機構によって，ソフトウェア開発者は同一の情報を複数のドキュメントに繰り返し記述する必要がなくなる．

3. 要素オブジェクトの表現モデルと Prolog による実装

要素オブジェクトの表現モデルでは，カスタマイズ機能を実現させるためにデータ独立性が重要となる．また，実装についてはデータベースのアクセス応答性とメモリ効率が重要である．

そこで，我々は表現モデルとして拡張フレーム表現を定義し，実装言語として Prolog を採用した．この選択によりフィルタなどをクラス情報から自動生成でき，生成した処理の応答性も高くすることができた．

3.1 拡張フレーム表現モデル

TENSE OMS の情報モデルでは個々のインスタンス，クラス要素オブジェクトが一個の拡張フレームにあたる．拡張フレームと通常のフレームとの相違点は第一に値以外に以下の情報カテゴリをカプセル化した点である．

- メタ属性：クラスの特性情報．書式情報等．
- メソド：フレームに付随する処理．フィルタ等．
- メタメソド：メソドの生成処理．

メタメソドはメタ属性を参照して各クラス固有のフィルタ等の処理を生成する．

第二の相違点は個々の拡張フレームがスロット以外にグループ，フレームと呼ぶ情報の内部階層をもつ点である．グループはフィルタ処理等で一括して扱うスロット群に関する情報をまとめた情報単位である．グループは自身に属するスロット名リストのほか，それらのスロットの内容を変更した場合影響を受けるクラス名のリスト等をメタ属性としてもつ．また，フレームは自分の所属する拡張フレームのグループ構成に関する管理情報等をメタ属性としてもつ情報単位である．

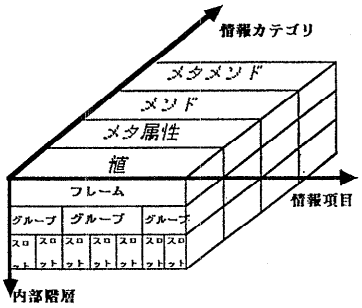

図 5 拡張フレームの構成
Fig. 5 Construction of extented frame.

以上をまとめると一個の拡張フレームは図 5 のように表すことができる．

3.2 Prolog による実装

TENSE OMS では格納や逆生成処理のように，一度に多種で大量の要素オブジェクトへのアクセスが頻繁に発生する．そこで，我々は実装用言語として Prolog を採用した．Prolog ではユニフィケーション機能により高速のデータアクセスを実現でき，言語の高い了解性によりメタメソド等のプログラムの保守性も高まる[12]．

Prolog による実装上の特徴は一個の拡張フレームの値やメタ属性情報をグループを最小単位とする複数の事実節に分割した点である．この事実節をアクセス単位と呼ぶ．特に，同一グループに属すスロットをまとめて一個のアクセス単位としたものをスロットグループと呼ぶ．このアクセス単位の導入により，相互に密接にからむメモリ効率，アクセス時間，フィルタ処理効率，保守性を実用的な範囲に納めることができる．

アクセス単位の形式は値，メタ属性で次のように共通の構造となる．網掛部分はオプションである．

```
ElementalObject(上位クラス名,グループ名,メタ属性名,
                識別情報,スロット1/メタ属性値,スロット2...).
```

ここで，"ElementalObject" はクラスではクラス名，インスタンスでは実際のモジュール名，変数名等の末尾に衝突を避けるため一個のスペースを付加したものである．また，ローカル識別情報は要素オブジェクト名と上位クラス名からアクセス単位が一意に決まらない場合必要になる情報である．たとえば，ローカル変数のローカル識別情報は所属する手続きの名称となる．

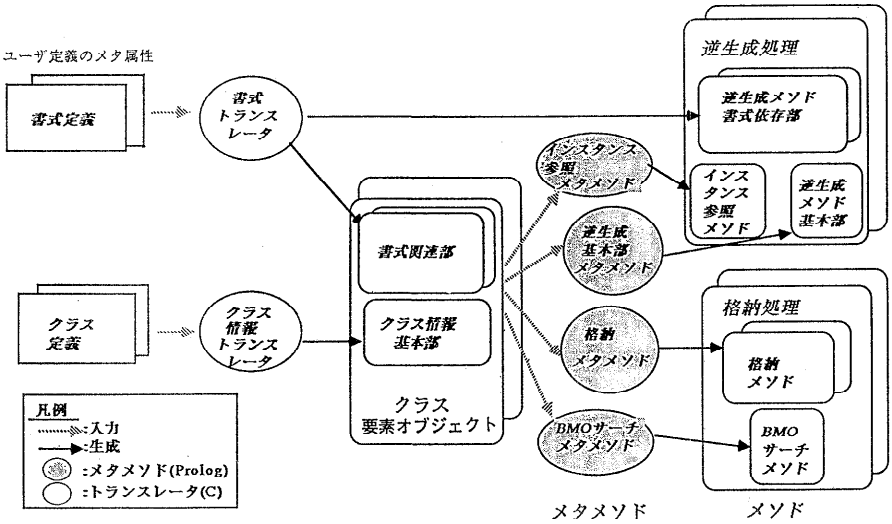

図 6 TENSE OMS のカスタマイズ機能
Fig. 6 Customization function of TENSE OMS.

4. カスタマイズの実際

ソフトウェアプロセスはソフトウェア開発プロジェクトの目標やチーム構成などによって多様だが，開発過程に現れるドキュメントに依存する部分が大きい．TENSE OMS を SEE に組み込むには，これらのドキュメントの書式とその記述内容に応じてクラス構成をカスタマイズする必要がある．また，SEE の進化に応じたカスタマイズも必要となる．

TENSE OMS のカスタマイズは次の三つの作業からなる．

- クラス定義：専用言語によるクラス構成の定義と，書式以外のメタ属性の定義．
- 書式定義：専用言語による各ドキュメント，各クラスごとの書式メタ属性の定義．
- 抽出処理の開発：各ドキュメントの抽出処理の開発（部分的にはクラス情報から自動生成する）．

クラスのアクセス単位はクラス定義と書式定義から各専用トランスレータによって生成する．このクラスのアクセス単位からメタメソドがフィルタ関連のほとんどのメソドを生成する．ただし，逆生成メソドの書式に依存する部分は書式情報から書式トランスレータで直接生成する（**図 6**）．なお，図 6 の BMO サーチメソドとは変更の影響解析処理の一部である．

以下，統合化Cプログラミングシステムの実現を例にとって，TENSE OMS のカスタマイズを具体的に述べる．

4.1 クラス定義

このステップではドキュメントの情報構成に基づき，クラス構成，クラス相互の汎化関係，各クラスのグループ，スロット構成，各スロットのデータタイプ，書式以外のメタ属性等を定義する．

クラス構成定義ではドキュメントの書式とC言語文法の非終端記号に基づいて末端のクラスを定義し，類似クラスの共通部分をくくって上位クラスを定義した．

複合関数クラスの定義例と定義から自動生成したアクセス単位例を**図 7** に示す．複合関数とはC言語の大域関数と static 関数の共通の情報から構成した AKO 上位のクラスである．図の定義部分の "storeReflect" の右側は変更の影響解析で探索すべきクラス名のリストである．また，定義部分の "basic" 以降は基本グループの定義である．原則としてすべてのクラスは一個の基本グループをもつ．なお，"#" で始まる単語はスロット名で，その右側は属性スロットの場合はデータタイプ，関係スロットの場合は関係の種類と相手のクラス名である．

4.2 書式定義

このステップでは各ドキュメントの書式を各クラスのグループごとに定義する．ソースプログラムのための関数クラスの基本グループの書式定義を**図 8** に示す．

この図で "src" とはソースプログラムの書式名である．マップ情報とはグループの各スロットが書式上に

```
/* 複合関数 の クラス定義 */
^synFunction:4,   11;   複合関数(synthesize)
     /* グループ数：4,スロット総数：11；複合関数クラス */
  AKO:^logical,[^function,^sFunction]
     /* 上位：論理情報，下位：関数，スタティック関数 */
                    :
  {basic  :7,基本  /* スロット数：7,基本グループの構成 */
     storeReflect:   [^dDesign,^mSource];
     (#caption  :%slist) /* タイトル：文字列リスト */
     (#explain  :%slist) /* 解説：文字列リスト */
     (#type     :%slist) /* 型名：文字列 */
                :int     /* 省略時値：整数値定義 */
                    :
     (#RELsynFunction:[REF,^synFunction])/* 関連関数 */}
```

```
%%クラス定義から生成した複合関数のクラス情報基本部
%  フレームメタ属性
%     akoメタ属性：AKO上下位クラス構成
       '^synFunction'('F^logical':ako,
       ['^logical',['^function','^sFunction'],synthesize]).
                    :
%     gmemメタ属性：グループ構成
       '^synFunction'('F^logical':gmem,[fdec,code,algo]).
%  グループメタ属性
%     smemメタ属性：グループ毎のスロット構成
       '^synFunction'('^logical_G':smem,
        ['caption','explain','type',...,'RELsynFunction']).
%     storeReflectメタ属性：定義変更時の伝搬対象
       '^synFunction'('^logical_G':storeReflect,
              [['^dDesign','^mSource'],[]]).
```

図 7 クラス定義とクラスアクセス単位の対応例
Fig. 7 An example of a class definition and generated class access units.

```
src{ /* 関数クラスの基本グループ用ソースプログラム書式定義 */
^function(,,[0,[#caption,...,-#RELsynFunction,#BMOfArgument]]){
body:{                                    /*マップ情報 */
     '/*>>> C言語関数 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<'.
     '/*      > ' #name-30 ':' #caption ' <'.
     '/*                    [ 説明 ]'.
                    :
     fout(#name_fdec,[#BMOfArgument,#type]).
     fout(BMOfArgument).
                    :
     '}/*>>>>>>>>>> "' #name '" 終了 <<<<<<<<<<<<<<<<<*/'.}}
```

図 8 書式定義の例
Fig. 8 An example of a part of document format definition.

現れるか否かを示している．この図では関連関数名を保持する"#RELsynFunction"スロットがソースプログラム上に現れないことを示している．

書式定義の本体は"body"以下である．"#"で始まる単語はスロット名で，スロットの内容をその出現部分に埋め込むことを表す．"fout"はその出現部分に他のグループやクラスの書式を埋め込むための関数である．このクラス名は引数の関係スロット名から導出する．

4.3 抽出処理の開発

統合化Cプログラミングシステムで扱うドキュメントは，詳細設計書とソースプログラムの二種類である．抽出処理はこれらのドキュメントから要素オブジェクトを抽出する処理である．

抽出処理は構文解析に基づくテキスト変換プログラムである．したがって，C言語の構文解析部分は再利用できるが，書式の構文解析部分とスロット情報の取り出し部分はドキュメントごとに開発する必要がある．取り出した情報はいったん抽出処理の内部テーブルに格納した後，抽出情報として出力する．この内部テーブルと出力処理はクラス定義より自動的に生成する．

抽出情報はグループごとに次のような形式となる．

:-上位クラス名_グループ名
　　(@ドキュメント書式名,
　　インスタンス名，スロット1の情報…)．

この形式は格納メソドを呼び出すPrologの質問形である．したがって，これをTENSEデータベースに読み込むと対応する格納メソドがただちに起動する．

なお，以上の抽出方式からTENSE OMSの扱うドキュメントは構文解析可能である必要がある．この要請から5章で述べるように構文からの逸脱を防止するツールも用意している．しかし，実際の運用ではユーザとの合意のうえ，条件コンパイル等は容易に構文解析できる範囲に記述を制限している．

4.4 メソドの生成例

クラス定義と書式定義からメタメソドによって各種メソドを生成できる．さきにあげた図7と図8の定義から自動生成した関数クラスの基本グループの格納メソドの処理は次のとおり（**図9**）．

i）抽出情報とデータベース内容の照合チェック．
ii）一致した場合は無為．
iii）食い違う場合はデータベース内容を更新．
iv）未定義の場合はそのまま格納．

```
/*************************************************************************
/*    "^function" 定義格納メソド  (自動反映：フェーズ)
/*************************************************************************/
'^function'('@src',Instance,Caption,...,_,BMOfArgument):- !,
  New =.. [Instance,'^function',Caption,BMOlocalPmacro,BMOfArgument],
  (call(New);
    (Old =.. [Instance,'^function',_,_,...,BMOlocalPmacro,_],
      (retract(Old),
      storeReflect('^function',Instance,[['^dDesign','^mSource'],[]]);
      true)),
    assert(New)).
/******************** "^function" END ******************************/
```

図 9 格納メソドの生成例
Fig. 9 An example of a generated accumulation method.

5. 統合化Cプログラミングシステムの機能と性能評価

以上によって TENSE OMS のカスタマイズは終了する．しかし，フィルタや各種情報サービスツールの運用方式については，ソフトウェア開発支援の事情に応じてアレンジすることができる．

統合化Cプログラミングシステムでは図 10 のように専用エディタとXウィンドウに基づくインタフェースを用意している．このエディタをフレームエディタ[15),18)]と呼ぶ．

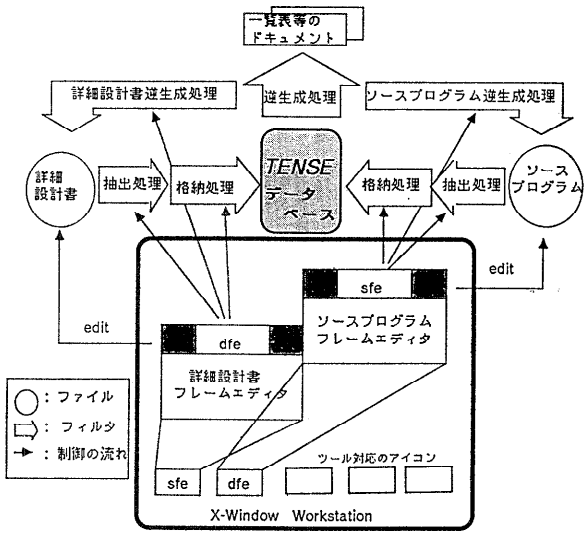

図 10 統合化Cプログラミングシステムの構成
Fig. 10 Overview of Integrated C Programming System.

```
------------------------------ 関数詳細設計 ------
file_to_list:情報ﾌｧｲﾙからﾒﾝﾊﾞの構造体のﾘｽﾄを作成
--------------------------------------------------
1) 機能
   ・概要
      1. ﾒﾝﾊﾞ情報のﾒﾝﾊﾞ情報構造体への格納.
2) リンケージ
   ・形式 - 型名付き関
   file_to_list    ()
   ・使用関数
     - 型名付き関数名
   talloc()        :
3) アルゴリズム
        1. .....
4) 関連関数
```

```
/*>>> C言語関数 <<< <<<<<<<<<<<<<<<<<<<<<<<<<<<<
/*  > file_to_list <
/*      -- 情報ﾌｧｲﾙからﾒﾝﾊﾞの構造体のﾘｽﾄを作成 --
/*                  [ 説明 ]
/* <機能>
/*  1. ﾒﾝﾊﾞ情報のﾒﾝﾊﾞ情報構造体への格納.
/*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
/*          [ 関数頭部 ]
/* >> タイプとパラメータ << */
  int file_to_list(){
/*          [ プログラム本体 ]
/* >> オートマティック変数 << */
  ;                 /* */          ...空欄
/* >> 使用関数 << */
  struct person *talloc();/* 記憶領域の動的確保 */
/* >> 実行コード << */
                                   ...空欄
}/*>>>>>>>>>> "file_to_list" 終了 <<<<<<<<<<<<<*/
```

図 11 ソースプログラムの部分生成例
Fig. 11 An example of a generated source program.

フレームエディタはユーザの書式から逸脱した記述を防止し，構文解析によって容易に要素オブジェクトを抽出できるようにする．また，書式の規則性を利用して編集作業を支援する．さらに，このエディタは編集作業の開始，終了時点でフィルタを自動的に起動する．したがって，統合化Cプログラミングシステムのユーザは TENSE OMS の存在を意識せずに，その提供する開発支援機能を享受することができる．

以下，統合化Cプログラミングシステムで実現した TENSE OMS の機能を概観し，応答性能等を評価する．

5.1 TENSE OMS に基づく開発支援機能

1) ソースプログラムの部分生成

詳細設計書格納の後，逆生成することにより部分的なソースプログラムを得る機能（図11）である．詳細設計書で記述のない情報は空欄となるので，フレームエディタによりこのソースプログラムを空欄記述式に完成させることができる．

2) ドキュメントの補完

他のモジュールの変数や関数等の参照記述を補完する機能である．前項の図 11 の例では，詳細設計書で記述のなかった使用関数 "talloc" のタイプとコメントをソースプログラムの生成時点で補完している．

3) 変更の自動反映

ドキュメントの変更によって影響の出る他のドキュメントに，変更内容を自動的に反映させる機能である．

格納時に影響解析が働き，影響を受けるドキュメントに通知を出す．通知を受けたドキュメントは次の編集開始時に逆生成処理によって変更の反映を受ける．

この機能の応用として，レヴィジョン番号を属性として詳細設計書とソースプログラムで共有させることにより両者のレヴィジョン番号の同期をとっている．

4) 静的解析

二つの機能がある．第一は TENSE データベースの内容を，直接静的に解析する機能で

表 1 フィルタ処理の real time (Sparc Station 2).
Tablel 1 A real time table of filters manipulation.

分類 / ファイル名	詳細設計書		ソースプログラム	
	mfile. d	mlist. d	mfile. c	mlist. c
サイズ(行)	315	283	359	220
抽出+格納	3.6	2.8	5.5	2.7
逆生成	1.6	1.2	2.6	1.7
コンパイル	—	—	2.6	1.7

ある.

第二は異なる時点でセーブした TENSE データベース同士を比較解析し，食い違いをレポートする機能である．この機能を TENSE OMS では，時差解析[16]と呼ぶ．時差解析の結果はウォークスルー，レビューに役立つ.

この時差情報を応用してデータベースの差分管理も実現している.

5) ドキュメント生成

TENSE データベースの内容を Tex 等のドキュメント形式で出力する機能である．各種一覧表，データベースの情報の蓄積度に基づく進捗度レポートなどを生成できる．逆生成処理の応用である.

5.2 統合化Cプログラミングシステムの性能評価

統合化Cプログラミングシステムではドキュメントの編集開始と終了時点にフィルタ処理がはいる．したがって，フィルタ処理の速度は応答性にとって重要である.

編集開始時点では編集対象ファイルが変更通知を受けている場合，自動的に逆生成処理が起動する．しかし，**表1**のとおり逆生成の所要時間は微小で，応答性はほとんど劣化しない.

編集終了時点では変更が皆無の場合を除き，毎回抽出，格納処理がはいる．表1のとおり所要時間はコンパイル時間の二倍相当である．しかし，これらの処理はバックグランドで行い，フォアグランドではただちに次の編集処理を始めることができるため応答性の問題は発生しない.

なお，格納処理時間は格納結果を save ファイルとして保存する所要時間も含んでいる．また，記憶容量については情報の重複する部分を一元化する分，個々のドキュメントの総和より減少する．しかし，アクセス単位に内部構造をもたせた分，格納前のテキストファイルより増加する．表1の例では4個の格納対象ファイルのサイズの総和は約 39 K バイト，格納後の TENSE データベースの増量は約 53 K バイトと，約1.4倍となっている.

6. 関連研究

以上，TENSE OMS について述べてきたが関連する研究として以下のようなものがある．これらの研究は目標やアプローチの方法で TENSE OMS と共通点がある.

OMS の研究として注目すべきものは Arcadia プロジェクト[1),3),4)]である．このプロジェクトは大規模コンソーシアムとして現在進行中である．目標は緊密に結合しながら柔軟で拡張性の高い SEE の実現である.

この環境は OMS，プロセスプログラミングインタプリータ，UIMS (User Interface Management System) からなる．同じ情報を何度も入力する必要のない環境の条件として OMS を重視する点が我々の目標と共通している.

日本における OMSの 研究には KyotoDB[5)]がある．この研究では製品オブジェクトと関係オブジェクトを分けて管理する方式をとる.

また，ソフトウェア情報をフレーム表現で扱おうとする研究には MicroScope[6)]がある．しかし，この研究の目標は知識ベース化した情報の静的な解析にある.

SEEの統合を推進するためにデータ管理を標準化しようとするプロジェクトに PCTE[8)] がある．TENSE OMS と同様 OMS を重視し，オブジェクト間の関係をリンクで表現する．しかし，PCTEの主なねらいは，情報の細粒度化による効果ではなく，CASE ツール間での情報交換の標準的基盤の提供にある.

AD/Cycle[7)]のような CASE ツールではレポジトリ[9)]をベースとして整合性の保証をめざすものがある．しかし，上流から情報の食い違いの発生しやすい下流までのきめ細かい整合性の保証や作業の自動化等については今後への課題が残っている．なお，多くのCASEツールの出力情報はオープンな定型テキストファイルであることから，この情報を抽出して TENSE データベースに格納することができる.

7. おわりに

TENSE OMS が立脚する情報モデルとソフトウェア開発支援機能について述べた.

情報モデルの特徴は細粒度の情報単位である要素オブジェクトの導入，拡張フレームによるデータ独立性の高い表現，Prolog による応答性の高い実装である.

この特徴により TENSE OMS はドキュメント間の情報の整合性の保証という所期の目標を達成した．また，この保証に基づく機械的作業の自動化により，情報の重複による弊害を排除した．さらに，多様なソフトウェアプロセスや SEE の進化に対応できるカスタマイズ機能を実現した．

このカスタマイズ機能によって実現した統合化Cプログラミングシステムは高い応答性を実現しており，ソフトウェア開発現場での試用も始まっている．このプロジェクトは20名以上のメンバで，TENSE OMS をプロセスごとに用意する方式を用いて数十万ステップのプログラムを開発している．なお，このプロジェクトでは CASE ツールの出力する構造化チャートとのインタフェースを備えることによって，上流フェーズの支援も一部実現している．

今後の課題としては，ユーザの試用結果のフィードバックと，要素オブジェクト形式を活用したソフトウェアの部品化，再利用がある．

謝辞 本論文の執筆にあたり，懇切なご指導をいただいた慶応義塾大学環境情報学部，有澤誠教授に深く感謝いたします．

参考文献

1) Taylor, R. N. et al.: Foundation for the Arcadia Environment Architecture, *Proc. of SDE 3*, pp. 1-13 (1988).
2) Penedo, M. H. et al.: Object Mangement Issues for Software Engineering Environments —Workshop Report—, *Proc. of SDE 3*, pp. 226-234 (1988).
3) Wileden, J. C. et al.: PGRAPHITE: An Experiment in Persistent Typed Object Management, *Proc. of SDE 3*, pp. 130-142 (1988).
4) Hudson, S. E.: The Cactis Project: Database Support for Software Environments, *IEEE Trans. Softw. Eng.*, Vol. 14, No. 6, pp. 709-719 (1988).
5) Matsumoto, Y. and Ajisaka, T.: A Data Model in the Software Project Database Kyoto-DB, *Advances in Software Science and Technology*, Vol. 2, pp. 103-122 (1990).
6) Ambras, J. and O'Day, V.: MicroScope: A Knowledge-Based Programming Environment, *IEEE Software*, Vol. 5, No. 3, pp. 50-58 (May 1988).
7) Matthews, R. W. and McGee, W. C.: Data Modeling for Software Development, *IBM Syst. J.*, Vol. 29, No. 2, pp. 228-235 (1990).
8) Thomas, I.: PCTE Interface: Supporting Tools in Software-Engineering Environments, *IEEE Software*, Vol. 6, No. 6, pp. 15-23 (Nov. 1989).
9) McClure, C.: *CASE is Software Automation*, Prentice-Hall (1989).
10) Minsky, M.: *A Framework for Representing Knowledge in the Psychology of Computer Vision*, Winston (1975).
11) Clocksin, W. F. and Mellish, C. S.: *Programming in Prolog*, Springer-Verlag (1981).
12) Li, D.: *A Prolog Database System*, Research Studies Press (1984).
13) Pressman, R. S.: *Software Engineering*, 2nd ed., p. 567, McGraw-Hill (1987).
14) 有澤：ソフトウェア工学，p. 213，岩波書店 (1988).
15) 原田(編)：構造エディタ，p. 198，共立出版 (1987).
16) 西岡，平田ほか：オブジェクトマネジメントシステムにおける時差解析，情報処理学会ソフトウェア工学研究報告，79-4 (1991).
17) 西岡，渡邊ほか：統合開発支援環境 TENSE のソフトウェアデータベース，情報処理学会アドバンスト・データベースシステムシンポジウム'90 予稿集，pp. 33-42 (1990).
18) 西岡，平田ほか：統合化Cプログラミングシステムの実現，情報処理学会 CASE 環境シンポジウム予稿集，pp. 81-88 (1989).
19) 渡邊，西岡：TENSE のオブジェクト管理システムのカスタマイズ，第 42 回情報処理学会全国大会論文集(5)，4S-11，pp. 331-332 (1991).

(平成4年2月14日受付)
(平成5年1月18日採録)

西岡 健自（正会員）

昭和25年生．昭和50年東京大学理学部地球物理学科卒業．同年横河電機株式会社入社．システムプログラム，プロセス集中制御システム関連ソフトウェア製品の開発に従事．昭和57年研究開発部門に移り，ソフトウェア開発支援環境，グラフィックユーザインタフェース等の研究開発に従事．特に開発環境の統合に関心を持つ．現在，同社オープンシステム研究所に勤務．ソフトウェア科学会会員．

平田陽一郎（正会員）

昭和33年生．昭和57年千葉工業大学工業経営学科卒業．現在横河電機株式会社オープンシステム研究所に所属し，主としてソフトウェア開発支援環境の研究開発に従事．

渡邊多恵子（正会員）

昭和40年生．平成元年津田塾大学学芸学部数学科卒業．同年横河電機株式会社入社．ソフトウェア開発支援環境の研究開発に従事．現在，同社オープンシステム研究所に勤務．