論　文

設計工程に合わせたビュー生成を可能にするソフトウェア文書管理手法の提案

手嶋　茂晴[†]　　荒木　円博[†]　　阿草　清滋[††]

Structued Software Documentation with Multi-View Schema Adaptng to Design Process

Shigeharu TESHIMA[†], Mitsuhiro ARAKI[†], and Kiyoshi AGUSA[††]

あらまし　ソフトウェアは設計仕様書，プログラム，テスト仕様書などの一連の技術文書であり，ソフトウェアの設計工程はこれらの文書を参照/作成する工程である．ソフトウェアの設計効率及び信頼性向上のためには，ソフトウェアの設計工程に適した技術文書の管理方式を確立する必要がある．そこで我々はソフトウェアの文書構造に注目した管理方式を考案し，それにもとづくソフトウェア文書管理システム *Liaison* を作成した．我々は，実在のソフトウェアをオブジェクト分析した結果からソフトウェアの文書構造は文書要素とその間のいくつかのソフトウェア設計固有の関係によって表現可能であるとの前提に立ち，オブジェクト分析により得られた関係をもとに文書構造を定義する．ソフトウェア文書管理システム *Liaison* は既存の文書を構造化文書に変換する機能と参照時に各設計工程で検索のキーとなる関係によって文書の表示ビューを構成する機能をもつ．*Liaison* を車載電子制御装置の設計に適用し，ソフトウェア設計での効果を確認した．

キーワード　ソフトウェア工学，ハイパテキスト，構造化文書，World-Wide-Web，組込みシステム

1. まえがき

ソフトウェアとは，要求仕様書，プログラム仕様書，ソースコード，マニュアルなどといった技術文書の総称であり，ソフトウェアの設計工程はこれらの技術文書を作成する工程である．ソフトウェア（以下ではソフトウェアの文書としての側面を強調して，ソフトウェア文書と呼ぶ）の中には設計情報が散在しており，設計の後工程や利用/保守工程において的確な参照を可能にし，ソフトウェアの設計効率及び信頼性向上のためには，ソフトウェアの設計工程に適した文書の管理方式を確立する必要がある．

一方ソフトウェア文書はその内容に加え，文書構造が極めて重要な意味をもつ．要求仕様書における “構造分析” やプログラムにおける “構造化プログラム” は対象となるシステム/プログラムのもつ構造をどのように文書に反映させるかについての手法である．設計の各工程ではその工程に適した文書構造に沿って文書を作成するが，その文書を他の工程で参照するとき，文書を作成するときに想定した文書構造は必ずしも十分なものではない．

ソフトウェア文書を工程全般で参照できるようにするには，ソフトウェア文書の構造や相互関係から必要な文書をたどることが可能でなければならない．具体的には以下の三つの作業が必要である．

（1） ソフトウェア文書の構造や相互関係を明らかにすること．

（2） ソフトウェア文書の構造や相互関係を表すビューを定義する手段を提供すること．

（3） 設計工程に合わせて，具体的なビューを構成すること．

我々は上記の作業をソフトウェア文書管理手法という形でまとめ，車載用制御プログラムや産業用ロボット制御プログラムなどの組込みシステムのソフトウェア文書に適用した．このうち（2）と（3）のビュー定義とビュー構成の部分をソフトウェア文書管理システム *Liaison* としてツール化した．

[†]（株）豊田中央研究所，愛知県
Toyota Central R&D Labs., Inc., Nagakute, Aichi-ken, 480–1192 Japan
[††] 名古屋大学情報メディア教育センター，名古屋市
Center for Information Media Studies, Nagoya University, Nagoya-shi, 464–8601 Japan

Liaison は組込みシステムのソフトウェア文書を対象に主に以下の三つの機能をもつシステムである．

- 既存の文書に対して必要な構造や相互関係を表す属性情報を付加し，構造化文書に変換する機能
- 属性情報をもとに文書の構造や相互関係をキーにして個別の文書及び文書要素を参照する機能
- 相互関係によって文書を再構成し，それをビューとして表示する機能

上記のいずれの機能も WWW フレームワークの中で実現した．

これまでのソフトウェアの文書管理の議論 [1] では管理対象の粒度をどのレベルに設定するかが管理手法の機能を特徴づける重要な項目と考えられてきた．我々はソフトウェア文書を構造化文書の形式で管理/保存することによって，設計ツールに対してそのツールが必要としている粒度の情報を提供することを可能にしている．バージョン管理やコンパイルではファイル単位の文書を用いる一方で，検索などでは構造化文書内部の構文を解析し，文書要素を取り出すことによって，プログラム構文レベルのオブジェクトを処理対象にすることができる．

ソフトウェア文書のデータ形式や設計ツール間のプロトコルなど広く設計環境としては，PCTE [2] が標準として ISO で規格されている．我々の手法も PCTE で提案されている開放型のリポジトリの考え方に従う．しかし，現実に設計者に使われているソフトウェア設計ツールで PCTE に対応するものは極めて少ない．これは PCTE が提案している概念は包括的であるため，単体ツールでの対応では PCTE の目指す環境を実現することが困難であることが主な原因である．また，PCTE の一部の機能は構造化文書や WWW など新たなデファクトスタンダードによって簡便なシステム化が可能となってきている．

2. 構造化文書を内部表現とするソフトウェア文書管理

2.1 ソフトウェア文書構造

ソフトウェア設計の現れる文書，例えば仕様検討書，関数機能仕様書，ソースプログラム，プログラム解説書，利用者マニュアルなどの個々の文書を個別文書，また個別文書の集合をソフトウェア文書と呼ぶ．ソフトウェア設計においては個々の文書を個別に取り扱うことはまれで，多くの場合いくつかの個別文書の関係の中で文書を作成/参照する．ここでは組込みシステム設計での関数設計仕様書とソースプログラムを例にソフトウェア文書の構造，つまり，どんな文書要素がどのような相互関係をもっているのかを明らかにする．設計工程のワークフローを観察/解析した結果，関数設計仕様書とソースプログラムに出現する文書の要素を表 1 に示すオブジェクトに整理できた．また文書中のオブジェクト間の相互関係としては一般の文書と同様，文書構成の階層関係の他にフトウェア文書に現れる関係には以下のものがあることが明らかになった．

- 詳細化/抽象化：設計工程の入力となる文書要素と出力となる文書要素の関係
- 更新：変更前と変更後の文書要素の関係
- バージョン：バージョンの異なるソフトウェアを設計する際，同じ位置付けにある個別文書間の関係
- データ依存（データフロー）：モジュール，関数などの間のデータフローの関係
- 制御構造（コントロールフロー）：モジュール，関数などの間のコントロールフローの関係
- 定義/利用：あるプログラム上のオブジェクトを定義している文書要素とそのオブジェクトを参照/利用している文書要素の関係（ソースプログラムでの変数や関数の定義とそれらの参照/呼出しの関係がこれに相当する）
- 派生/引用：オブジェクトの記述内容を他の要素から引用しているときの原典となるオブジェクト文書要素と引用しているオブジェクトの関係（仕様書にある表現をソースプログラムでコメントとして引用する関係などがこれに相当する）

ここに示す関係はソースコードの解析などで利用されるものと同様であるが，我々はこれらをソースコード以外のソフトウェア文書の解析にも適用する．表 1 に示すオブジェクトは組込みシステムに限られたものを含むが，オブジェクト間の相互関係については汎用的な上記の関係が適用できる．表 1 に示す文書オブジェクトとその間の関係を OMT 図 [3] に表現したものを図 1 に示す．図 1 では文書要素の階層関係をオブジェクト集約関係で，また派生/引用の関係をオブジェクト継承関係を用いて表現している．

2.2 文書管理と参照のための 3 層スキーマ

前項で明らかにしたソフトウェア文書の構造に基づいて，文書を管理/参照するためのデータベースのスキーマを提案する．我々の提案するスキーマは以下の 3 層から構成される．

- 構造参照スキーマ [Structural View]（外部ス

表 1　ソフトウェア文書における代表的なオブジェクトとその属性
Table 1　Major objects in software documents and their attributes.

個別文書	オブジェクト	主な属性
関数機能仕様書	項目	ファイル，バージョン，関数（ソース）
	状態量	データ型，範囲（上下限），変数
	処理	表現形式
	マップ表	軸
	数式	パラメータ，返値
	手続き表現	表現形式
ソースコード	大域変数定義	状態量
	関数	仕様項目，バージョン
	処理ブロック	処理

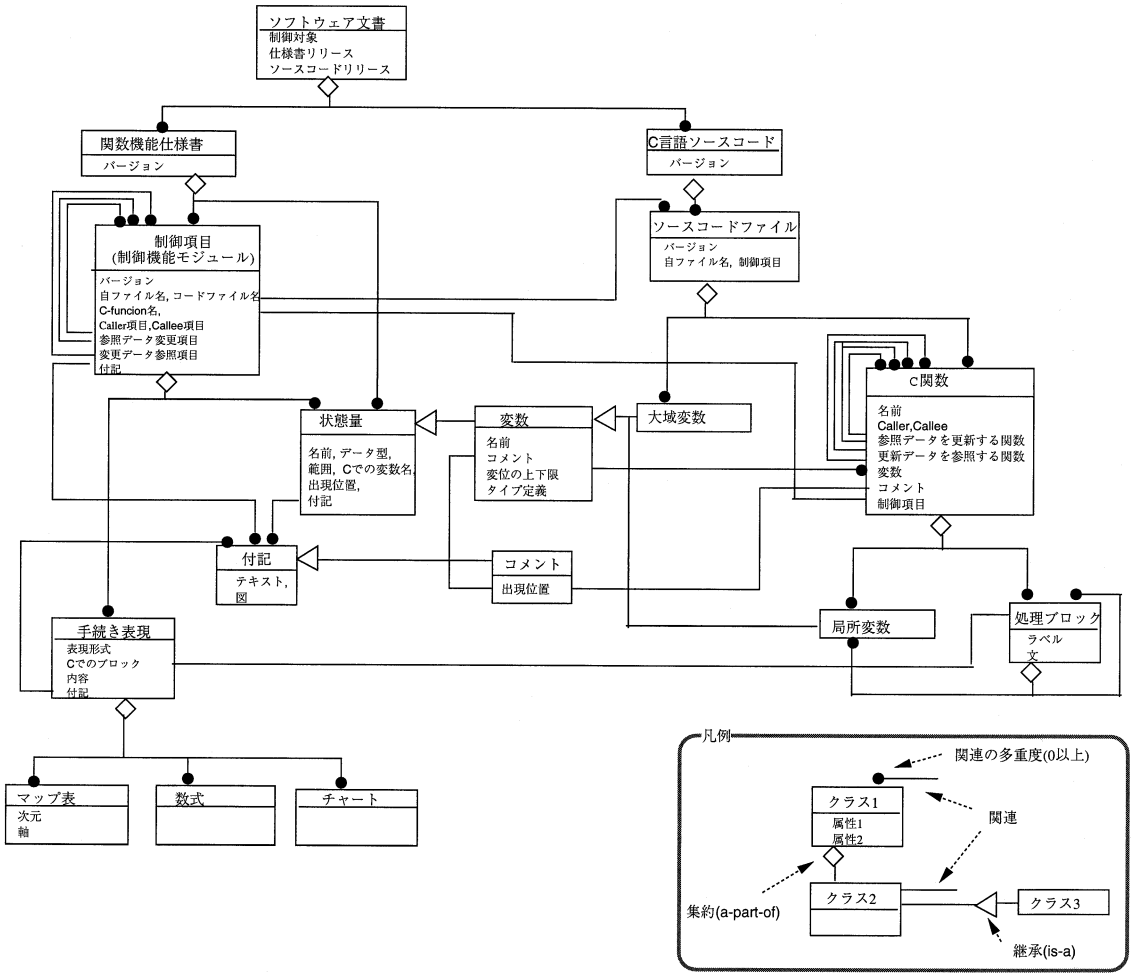

図 1　文書オブジェクトとその間の相互関係
Fig. 1　Document objects and their relationship.

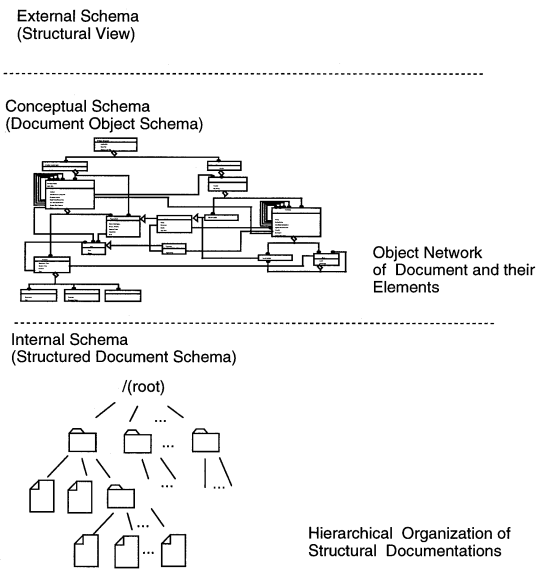

図 2　3 層スキーマ
Fig. 2　Three layered schema.

キーマ [External Schema]）

- 文書オブジェクトスキーマ [Document Object Schema]（概念スキーマ [Conceptual Schema]）
- 構造化文書スキーマ [Structured Document Schema]（内部スキーマ [Internal Schema]）

構造参照スキーマは文書を参照する利用者から見た文書の構造である．工程ごとに必要となる文書及び文書要素，関係をビューとして提供する．文書オブジェクトスキーマは文書のもつ相互関係を網羅（らら）するモデルである．このスキーマでは管理対象のソフトウェア文書をオブジェクトに分解しその間の関係を表す．前節で述べた文書オブジェクトとその間の関係をデータベーススキーマとしたものが文書オブジェクトスキーマである．構造化文書スキーマは概念スキーマである文書オブジェクトスキーマを構造化文書で内部表現するための内部スキーマである．これら 3 層のスキーマの構造を図 2 に示す．

構造参照スキーマで提供するものは，概念スキーマで定義された文書の関係のうち必要なもの又は演繹（えんえき）可能なものについて設計工程に適するように再構成された文書のビューである．例えばモジュール機能名をキーとする関数機能仕様書のリスト，版管理された文書要素の版数による木構造，また必要な関係をリンクとして埋め込んだハイパテキストに変換された文書などが構造参照スキーマのビューにあたる．

ソフトウェア設計者から見える文書構造である構造参照スキーマと概念スキーマである文書オブジェクトスキーマとを区別し，構造参照スキーマを概念スキーマの構成要素から再構成する手段を提供することによって，設計工程ごとに必要な文書のビュー生成が可能となる．

構造化文書スキーマは概念スキーマから代表的な集約関係によって作られた極大木（spanning tree）から構成される．極大木をあるレベルのノード（オブジェクト）で根側と葉（終端）側に分割する(注 1)．その分割によって以下に示すように構造化文書を再構成する．

（1） 終端側の文書オブジェクトは，構造化文書の文書要素とする（このようなオブジェクトを構造化文書スキーマにおいて仮想要素と呼ぶ）．

（2） 境界にある文書オブジェクトについては，そのレベルより終端側の文書オブジェクト（仮想要素）を文書要素にもつ構造化文書とする．

（3） 根側にあるオブジェクトはそれ単独で一つの構造化文書を構成させる．

図 3 に文書オブジェクトスキーマから構造化文書スキーマに変換される例を示す．図 3 において太線によって極大木は表現されている．ClassF,G,H,I,J のレベルで根側と葉（終端）側に分割している．

構造化文書スキーマは上記の（2）の文書を終端に，（3）の文書を非終端にもつ木構造で表現される．文書オブジェクトスキーマにおける文書オブジェクト間の関係は文書要素の属性（Attribute）の形で表現する．単一粒度のデータベースシステムや版管理の既存のリポジトリシステムによって構造化文書スキーマを実現するときは，構造化文書とその木構造をそのリポジトリ内部形式で管理する．構造化文書内部にある細粒度の文書オブジェクトについては既存リポジトリシステムの管理対象に含めない．構造化文書の内部構造は，リポジトリシステムとは別に文書オブジェクトスキーマと構造化文書スキーマの変換機構で管理する．

本論文の管理手法と，既存の単一粒度のリポジトリシステムを適用する手法 [4], [5] との比較を表 2 にまとめる．ここでの比較では典型的な管理粒度（ファイルとプログラム構文要素）の場合を示している．

(注 1)：どこで分割するかは，システムを実現する上での制約はないが，実用上は実際の一つのファイルで表現されている程度の粒度で分割するのが適当である．

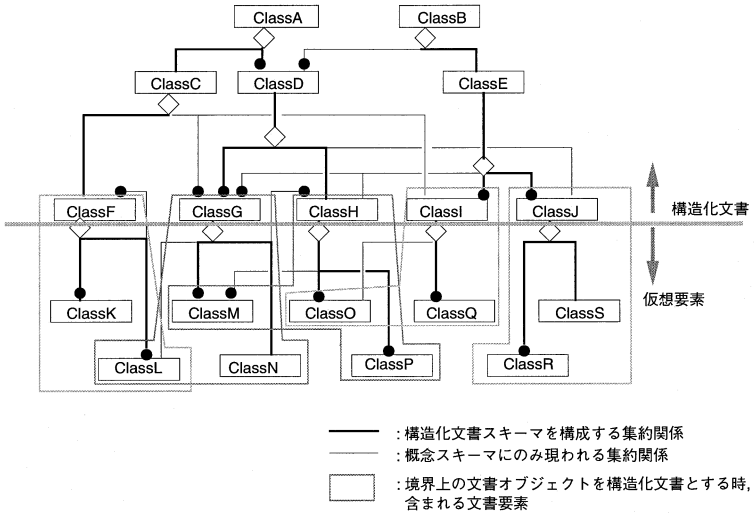

図 3　構造化文書スキーマへの変換
Fig. 3　Transforming to structured document schema.

表 2　提案管理手法の特徴
Table 2　Characteristics of our method.

	操作		
	製品パッケージ管理	ファイル処理	構文要素での検索/ 更新
リポジトリ単独（粒度：ファイル）	版情報を別ファイルで管理（一貫性保持が必要）	○	文字列操作で代用（精度が低い）
（粒度：構文要素）	–	要素からファイルを再構成（設計者の参照に不適）	○
提案手法	外部スキーマとして実現	構造化文書から容易に必要な形式に変換	構造化文書を構文解析した上で機能を提供

3. ソフトウェア文書管理システム *Liaison*

3.1 *Liaison* での文書構造

我々は，前章で述べた文書管理手法に基づく *Liaison* を開発している．*Liaison* では構造化文書スキーマの構成要素となるノード（構造化文書）を SGML [10] 形式の構造化文書とそれらの木構造を表す属性によって実現する．

つまり構造化文書スキーマでのノードを内部データでは，

- 識別子（ID）
- 構造化文書表現

のデータと

- 親ノード，子ノード
- その他必要な関係をもつノード

へのリンクによって表現する．

ここで識別子はソフトウェア文書データベース全体で文書要素を識別するための ID である．*Liaison* では文書要素に構造化文書スキーマの木構造と構造化文書の階層構造に従うパスによって決まる一意の ID を与える．具体的には集約関係においては文書要素に重複を許さないキーを与えることとし，そのキーを構造化文書スキーマで決まるパスに従い "/" でつないだ文字列を ID とする．例えば，図 3 にある構造化文書スキーマの場合，ClassS のインスタンスの ID は，"`/key-B/key-E/key-J/key-S`" となる．ただし，ここでは ClassS のインスタンス自身のキーを `key-S`，そのインスタンスを包含する ClassB,E,J のインスタンスのキーを `key-B`，`key-E`，`key-J` としている．

構造化文書表現は SGML のインスタンスであり，SGML のタグ及び属性によって以下の情報を表現する．

- このノードの文書としての内容とその論理的な

レイアウト情報

- このノードがもつ仮想要素のオブジェクトとしての属性（クラス及び関係）

親ノード及び子ノードは，構造化文書スキーマの木構造での親/子ノードの ID である．構造化文書表現を解析すれば，この親及び子ノードの ID は取得可能であるが，親及び子ノードの ID は頻繁に参照されるので構造化文書表現とは別にもつことにする．

その他必要な関係をもつノードも親及び子ノードと同様で，そのノードから頻繁に参照されるノードの ID を関係名と対にしてノードの内部表現としてもつ．

3.2 システム構成

Liaison は WWW をプラットホームとするシステム上に実現され，WWW サーバを介して利用者が操作する WWW クライアントで文書を表示させるシステムである．システム構成を図 4 に示す．

Liaison は文章の参照に先立ち文書をシステムに取り込む “登録フェーズ（Regstration Phase)” と設計工程においてソフトウェア設計者からの参照要求にこたえてビューを提供する “参照フェーズ（Reference Phase)” に分かれる（図 5 参照）．

登録フェーズにおいては既存の構造化されていない文書を *Liaison* の内部データに変換するツールを用意している．このツールは登録対象の文書を構造化文書スキーマで定めた SGML インスタンスの木に変換する．

また，ソースプログラムについては ANSI C 言語プログラムを SGML へ変換するツールを用意している．このツールは関数機能仕様書との関係付けに必要な構文要素を認識し，その出現位置などを内部データ形式

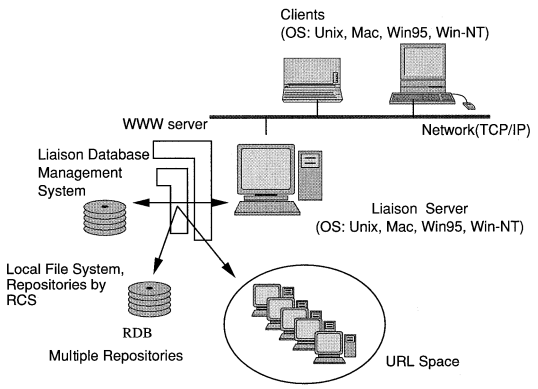

図 4　システム構成
Fig. 4　Structure of *Liaison* system.

に変換する．変換のための C ソースプログラムの構文の解析には Sapid（Sophisticated APIs for CASE tool Development）[8] を利用している．

一方，参照フェーズの *Liaison* は前章で述べた 3 層のスキーマ構造に基づいた参照を実現するために各層に対応した以下の三つ機能を提供する．

（1） 利用者が工程別に用意するビューを定義するテンプレート

（2） 文書オブジェクトスキーマで定義されるオブジェクト（文書要素）へのアクセス関数及びその内容を HTML 形式へ整形する関数

（3） 構造化文書スキーマを構成する要素をリポジトリから獲得する関数

Liaison ではそれぞれの部分を以下のように実現する．（1）の部分は WWW サーバで実行されるスクリプト言語のプログラムとして書き下され，サーバ上では CGI（Common Gateway Interface）に従う関数として実装する．ビュー定義のために必要な文書オブジェクトへのアクセスは（2）のアクセス関数を介して行う．アクセス関数の返り値を HTML の文法に合わせてレイアウトすることによりビューを構成する．

（2）及び（3）の関数は WWW サーバから起動される Perl プログラム群の形で実装されている．Perl の各プログラムは URL 形式の文書要素へのアクセス要求を解釈し，対応する要素をクライアントへ返す機能をもつ．このうち（2）の部分について，関数の処理内容は文書オブジェクトスキーマによって決まるものである．また（3）の部分については，構造化文書の格納するリポジトリ別に準備する．（2）のアクセス関数の詳細については次節で述べる．

3.3 文書データベースへのアクセス手段

ビュー設計者に対して *Liaison* は文書データベースへのアクセス手段を CGI 形式で提供するパッケージである．前節で述べた文書スキーマへのアクセス関数はビュー設計者から見ると，次に示す 3 階層で実現されている．

- 第 1 階層：識別子と属性名から文書オブジェクトの属性値を返す関数（内部スキーマに対応）
- 第 2 階層：文書の構造や相互関係を関数や属性に関しての条件を満たす文書オブジェクトを返す関数，及び文書要素の内容と構造を HTML 形式に変換して返す関数（概念スキーマに対応）
- 第 3 階層：構造参照スキーマであらかじめ定められたビューに従い，文書要素の表現を加工し，その

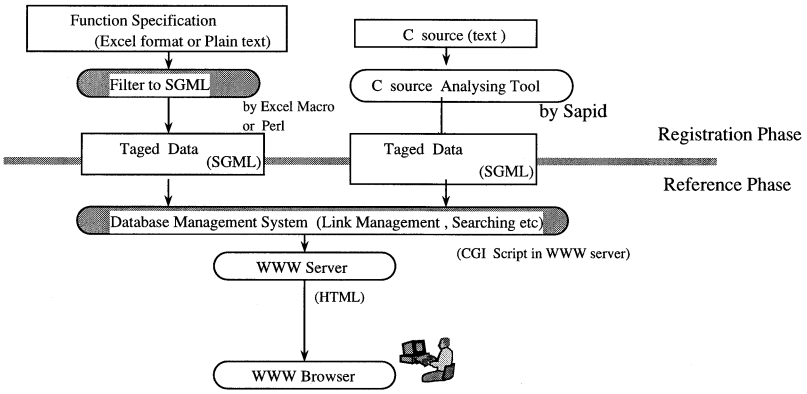

図 5 処理手順
Fig. 5 Procedual flow.

結果を HTML 形式に変換する関数（外部スキーマに対応）

第 1 階層のアクセス関数は，まず文書オブジェクトの識別子からその文書オブジェクトを収めたリポジトリへのアクセス手段（ファイル名，コマンド系列，データベース質問式など）からその文書オブジェクトを含む構造化文書を獲得する．その次に対象の文書オブジェクトが構造化文書スキーマにおいて仮想要素である場合は，それを含む構造化文書表現を解析して必要な情報を取り出す．このアクセス関数はアクセス対象が構造化文書スキーマでどのように表現されているかを隠蔽する．第 2 階層のアクセス関数は第 1 階層のアクセス関数の組合せで実現される関数群である．第 3 階層は WWW ブラウザでの直接の利用を想定したもので，第 1 及び第 2 階層のアクセス関数の結果を HTML 形式で返す．

4. 適用事例及び評価

4.1 適用対象工程

車載用制御プログラムに *Liaison* を適用した結果を示す．

車載用制御プログラムの開発は，プロトタイプによる先行開発を受けて以下の工程を経て最終製品プログラムとなる．

（1） 関数機能仕様書作成：先行開発で得られた制御アルゴリズムをプログラム化できるように手続き的に記述する．

（2） 基準プログラム作成：機能仕様書の内容を反映したプログラムを作成する．

（3） 適合：車両/エンジンの細かな種別に合わせて制御パラメータを最適化する．同時に最適化の結果を反映するよう関数機能仕様書を更新する．

我々が今回 *Liaison* を適用したソフトウェア開発工程は以下の二つである．

- 適用事例 1: エンジン燃料噴射制御システム [13] の基準プログラム開発工程
- 適用事例 2: 車両横滑べり防止システム [14] の適合工程

この 2 例では構造化文書スキーマとビューの構成が異なる．これらの違いから我々の提案する管理手法の特徴を評価する．

4.2 システム構成及びビュー定義

適用事例 1，2 ともそれぞれの工程での作業内容と既存の文書管理の形態と整合性を維持するようにシステムを構成した．対象の文書は関数機能仕様書と C 言語ソースプログラムである．

適用事例 1 では以下の機能を実現する．

- 既存の文書管理形態を踏襲し，ファイル単位での操作については既存システムとの互換性を確保する．具体的には OS ファイルシステムでのディレクトリ及びファイル名によるバージョン変更履歴管理(注 2) を行う．
- SGML 形式の関数機能仕様書と C 言語ソースプログラム間の構文要素レベルでの対応付け（構文要素レベル粒度での管理）を行う．

(注 2)：例えば，C ソースコードファイル 111.c に対して以後の更新されたバージョンをファイル 111a.c, 111b.c というファイル名で作成する．

既存管理形態との整合性維持のため図 1 に示した概念スキーマで関数機能仕様書と C ソースプログラムファイルのレベルで構造化文書を構成した．関数機能仕様書については既存の文書を SGML 表現に変換したものを用いた．図 1 の概念スキーマの関数機能仕様書以下にあるクラスに対応する文書要素を DTD で定義している．図 6 に関数機能仕様書の SGML 表現の例を示す．C ソースプログラムファイルについては表現はそのままにし，相互参照に必要な情報を構文解析した結果を利用する．

設計者に対しては以下などのビューを作成し，WWW で表示できるようにした．

- 制御項目順の関数機能仕様書リスト及びリストから関数機能仕様書へのリンク（図 7 左部分参照）
- 関数機能仕様書からバージョンの前後の版へのリンクとそれを実現した C 言語ソースプログラムへのリンク（図 7 右部分参照）
- 関数機能仕様書中でのプログラム構文要素について説明箇所からそのプログラム構文要素の C 言語ソースプログラム中での定義位置へのリンク（図 8 参照）

```
<!DOCTYPE ECU-SPEC-FILE SYSTEM
 "ECU-SPEC-FILE.DTD">
<ECU-SPEC-ITEM
  NAME="クランク角信号の処理"
  >
   ...
 <SECTION NAME="グローバル変数">
  <DECLARATION
   NAME="crank_count"
   DATA-TYPE="char"
   SCOPE="whole"
   ...
   >
 <SECTION NAME="スタティックフラグ">
  <DECLARATION
   NAME="crank_flag"
   SCOPE="whole"
   ...
   >
   ...
 <SECTION NAME="手続き">
  <CHARTS
   <SP-BLOCK>
   ...
   >
   ...
</ECU-SPEC-ITEM>
```

図 6　適用事例 1 で用いた関数機能仕様書の記述例
Fig. 6　Sample description of function specification in case 1.

事例 2 は事例 1 と以下の点が異なる構成とした．

- OS ファイルシステムでのディレクトリによるバージョン管理と RCS [16] によるバージョン変更履歴管理の共存
- 関数機能仕様書と C 言語ソースプログラム間のファイル単位での対応付け

図 9 に関数機能仕様書の SGML 表現の例を示す．この記述では文書の段落を最小の文書オブジェクトとしている．各段落はソースプログラムとの関係を表すような属性はもたない．

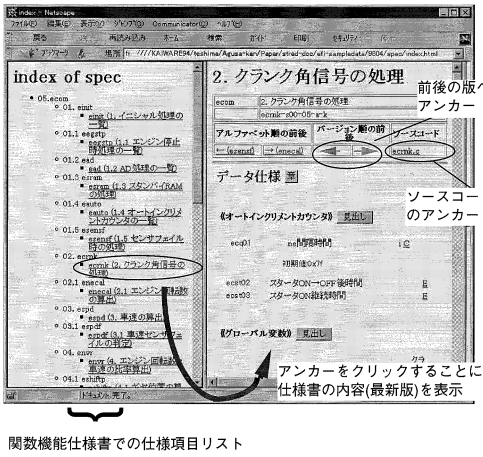

図 7　関数機能仕様書の制御項目リストの表示
Fig. 7　View of control items in functional specification.

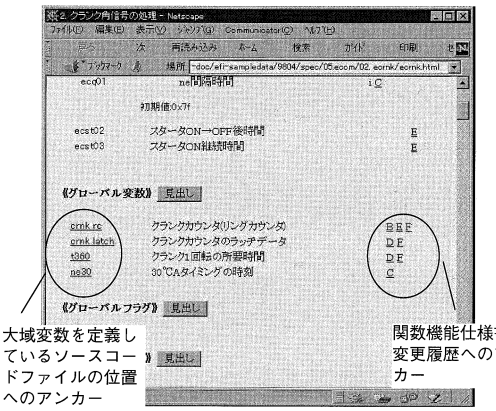

図 8　関数機能仕様書表示における関係リンクの実現
Fig. 8　Links embedded in view of functional specification.

設計者に対しては，以下などのビューを提供した．

- 車両バージョンごとの関数機能仕様書と C 言語ソースプログラムのリスト，及び各リストの項目からファイルへのリンク
- 関数機能仕様書と C 言語ソースプログラムの改訂リスト及びリストから該当ファイルへのリンク（図 10 参照）
- 関数機能仕様書及び C 言語ソースプログラムに対してソースプログラム中の大域変数のデータ依存関係により，大域変数の更新の影響を示すリンク
- 検索範囲を指定し，キーワードを含み文書の一覧及びその文書の内容を得るリンク（全文検索）

4.3 試行システムによる管理手法の評価

前節 **4.2** での二つのシステム構成で数か月間それぞれの設計工程で試行し，WWW サーバでの参照履歴や利用者（プログラム設計者）からの聞き取り調査をもとに以下の項目について評価する．

［*Liaison* システムの処理能力］

表 3 に，事例 1，2 の試行期間中に *Liaison* システムを実現したハードウェア/ソフトウェアの諸元を示す．表 4 に，試行期間中に *Liaison* に登録され，*Liaison* の管理化に置かれているバージョン数及びバージョン当りの文書の数/容量を示す．文書数は試行期間中にほぼ増加が停止し，それぞれの工程で参照が必要な文書をシステム管理化においたことが確認された．処理速度については CPU 性能が劣る事例 2 の環境で測定した結果を表 5 に示す．測定はクライアント数 1 の状況で，リンクとして用意されたビュー表示にかかる時間と，全文検索の時間を測定した．利用者からの聞き取り調査を行った結果，事例 2 の場合でも許容可能な応答速度であった．

```
<!DOCTYPE ECU-SPEC-FILE SYSTEM
 "ECU-SPEC-FILE.DTD">
<ECU-SPEC-FILE
  NAME="車輪速度推定演算">
  C-SOURCE=v-weel.c">
  >
 <SECTION NAME="演算開始条件">
  ...
 <SECTION NAME="車輪条件">
  ...
</ECU-SPEC-FILE>
```

図 9 適用事例 2 で用いた関数機能仕様書の記述例
Fig. 9 Sample description of function specification in case 2.

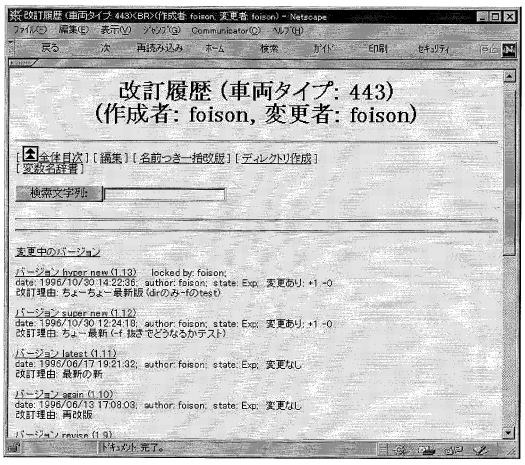

図 10 関数機能仕様書の更新ログ表示
Fig. 10 View of update log for functional specification.

表 3 適用事例でのシステム諸元
Table 3 System specification in case studies.

（a） 適用事例 1

ハードウェア		Sun Sparc10
ソフトウェア	OS	Solaris2.5
	httpd	CERN httpd

（b） 適用事例 2

ハードウェア		PC 互換機（pentium 75 MHz）（メモリ 32 MBytes, ディスク 500 MBytes）
ソフトウェア	OS	Windows NT 3.5
	httpd	WebSite（O'Reilly& Associates）

表 4 適用事例での管理対象文書数
Table 4 Number of documents in case studies.

（a） 適用事例 1

バージョン数	文書	ファイル数	ファイル平均行数
2	関数機能仕様書	180	500
	ソースプログラム	200	300

（b） 適用事例 2

バージョン数	文書	ファイル数	ファイル平均行数
4	関数機能仕様書	90	100
	ソースプログラム	300	130

表 5 適用事例 2 でのシステム応答時間
Table 5 System response time in case study 2.

指定ファイルの内容表示	仕様項目のリスト更新	プログラム全文検索（指定バージョン）
5sec	7sec	40sec

表 6　*Liaison* システムの機能別コードサイズ割合
Table 6　Code size ratio for each feature of *Liaison*.

ビュー定義　(35%)			文書オブジェクトアクセス (10%)		リポジトリアクセス (35%)			その他
集約関係	変更履歴	その他	（オブジェクト構造依存）	（オブジェクト構造非依存）	OS ファイルシステム	RSC	MS Access	（共通部分）
15%	5%	15%	8%	2%	10%	20%	5%	20%

［文書オブジェクトスキーマ及び構造化文書の導入効果］

今回提案する手法の最大の特徴は，プログラム構文レベルの粒度とファイルレベルの粒度が共存する概念スキーマを導入し，構造化文書を用いてそれを表現する点にある．事例 2 では概念スキーマにあたる文書オブジェクトスキーマのうち従来からの管理手法と同様にファイル単位の対応関係をビューとして実現した．事例 1 では構文要素での対応関係を含めて，概念スキーマで規定する関係を実現した．WWW サーバでの参照履歴を解析した結果，事例 2 では参照要求の約 20%がアンカーをクリックすることによって対応関係をたどるもので，残りは全文検索を利用して必要な文書を探す要求であった．事例 1 ではビューに定義されたアンカーをたどることでほぼすべての参照を処理していることがわかった．利用者からの聞き取りを併用した調査では，全文検索で検索結果から候補を絞り込むには概念スキーマでの文書属性を合わせて表示する必要があること，加えて，そのような対策を施しても 30%程度の割合で必要な文書を発見できない場合があることがわかった．また，構文要素での対応関係によるアンカーが提供された場合のほうが，ファイル単位の対応関係だけの場合と比べ参照したい文書若しくは文書要素を設計者の少ない操作で得られていることもわかった．以上からソフトウェア文書の管理では我々が提案する概念スキーマに従う参照が従来のファイル単位での管理と比べ有効であることがわかった．

［3 層スキーマの効果］

Liaison システムの 3 層スキーマ構造に対応する機能別のコードサイズ割合を表 6 に示す．表 6 から文書の内部表現を管理するリポジトリの変更や新たなビュー追加に少ないコード追加で対応できていることがわかる．これは *Liaison* が 3 層構造のスキーマに従う構造を採用することによって，モジュール性の高いシステム構成になっているためである．

5. むすび

ソフトウェア設計工程で用いる技術文書を対象とした文書管理方式を提案した．ソフトウェア文書は構成要素間に複雑な相互関係をもつ文書である．我々は文書中に現れるオブジェクトとその属性を分析することによって，ソフトウェア文書の構造を明らかにした．また，その解析結果をもとに，物理的なデータ構成と独立にデータベースシステムが構築できるような 3 層からなる文書データベースのスキーマを考案した．このスキーマを採用することによって，ソフトウェア文書管理システム *Liaison* は既存のデータリポジトリを利用しながら，文書構造に従った検索/参照を可能にしている．

ソフトウェア文書管理システム *Liaison* を車載用制御プログラムの設計工程に適用し，効果を示した．適用事例では我々の提案する文書管理方式によって設計工程に適した文書参照が簡便に実現可能となることで，管理上の以下の利点が確認できた．

- 文書作成/保守の作業工数の低減：制御用プログラムにかかわる技術文書は頻繁に改訂される．そのため，これまでの紙の『冊子』による文書管理には作成/保守に多くの工数がかけられている．文書の変更を瞬時に反映させ，参照することができ，文書の保守性が向上した．
- 検索性の大幅な向上：検索操作のユーザインタフェースの利便性と検索時間の短縮，検索対象の拡大によって，これまで時間がかかるために，調べることが実質的に不可能であった文書を容易に検索することができるようになった．
- 設計環境の統合：利用者へのビューの中に，設計ツールの起動/データの引渡しを埋め込むことによって，設計ツールの統一的なユーザインタフェースを提供できた．

現在のところソフトウェア文書管理システム *Liaison* で対象にしている文書は電子化を前提に作成されたものである．今後は，製品設計全般の文書を対象とできるように機能拡張を目指している．例えば，実験データや製品/部品仕様書などの図形的な情報を多く含む文書などを取り込むことによって保守工程までの製品

設計/製造の全般での利用を可能にする文書構造のモデル化とワークフローとの統合が課題である.

謝辞 ソフトウェア開発現場の実情の見学・聞き取りやプロトタイプについての議論に協力いただいたアイシン精機株式会社信頼性技術部河合種市 GL，同走行系技術部浅野憲司担当員,(株) デンソーロボット技術部小山俊彦担当部員，富士通テン株式会社モートロニクス本部開発部池添朗技師，斗納宏敏技師に感謝します．日ごろから御議論いただく（株）豊田中央研究所ソフトウェア研究室山崎知彦室長，佐野範佳研究員をはじめソフトウェア研究室の諸氏に感謝致します．また，愛知県立大学山本晋一郎助教授及び名古屋大学情報メディア教育センター濱口毅先生には数々の御助言をいただいたことを感謝致します.

文　献

[1] D. Schefström and G. Broek, "Environment Integration: Concept and Studies," Atmosphere deliverable D2.3.1.2.2-1.0, 1992.
[2] L. Wakeman and J. Jowett, "PCTE: The Standard for Open Repositories," PIMB Association, 1993.
[3] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design," Prentice Hall, 1991.
[4] Y. Chen, M.Y. Nishimoto, and C.V. Ramamoorthy, "The C Information Abstruction System," IEEE Trans. Software Eng., vol.16, no.3, pp.325–334, 1990.
[5] 鰺坂恒夫，松本吉弘，"ソフトウェアエンジニアリング・データベース KyotoDB の設計と実現," 情報処理学会論文誌，vol.33, no.11, pp.1402–1413, 1992.
[6] H. Ziv and L.J. Osterweil, "Research issues in the intersection of hypertext and software development environments," LNCS 896, pp268–279, Springer-Verlag 1995.
[7] K. Tanaka, N. Nishikawa, S. Hirayama, and K. Nanba, "Query pairs as hypertext links," Proc. of IEEE Data Engineering Conf., pp.456–463, 1991.
[8] 福安直樹，山本晋一郎，阿草清滋，"細粒度リポジトリに基づいた CASE ツール・プラットフォーム Sapid," 情処学論，vol.39, no.6, pp.1990–1998, 1998.
[9] 古館丈裕，岡安光彦，石川佳治，植村俊亮，"構造化文書データベースに対するラッピング手法の提案," 情処 DBS 技報 vol.96, no.68, pp.305–310, 1996.
[10] Martin Bryan 著，山崎俊一監訳，"SGML 入門," アスキー出版局，1991.
[11] S. Henninger, "Supporting the Construction and Evolution of Component Repositiory," 18th ICSE, pp.279–288, 1996.
[12] 関　良明，"分散型ノウハウ蓄積システム GoldFISH における分散環境への適応," 情処学論，vol.36, no.6, pp.1359–1366, June 1995.
[13] 菅沼英明，佐藤浩司，"自動車における組込みシステム開発の現状 — エンジン制御を中心として," 情報処理，vol.38, no.10, pp.892–897, 1997.
[14] 松田俊郎，"ABS の最新実用知識<その 1>," 自動車工学，pp.31–54, Sept. 1990.
[15] J. Nielsen, HYPER Text & Hyper Media, Academic Press, 1990.
[16] W.F. Tichy, "RCS—A System for Version Control," Software—Practice and Experience, vol.15, no.7, pp.637–654, 1985.
[17] D.J. Hatley and I.A. Pirbhai, "Strategies for Real-Time System Specification," Dorset House, 1988.

（平成 10 年 4 月 21 日受付，12 月 4 日再受付）

手嶋　茂晴　（正員）

1986 京大大学院工学研究科情報工学専攻修士課程了．1998 名大大学院工学研究科博士課程後期課程情報工学専攻了．1986 より（株）豊田中央研究所に勤務．1993 から 94 までスタンフォード大学客員研究員.

荒木　円博

1985 より（株）豊田中央研究所に勤務．ACM, IEEE-CS, ICMA, 情報処理学会，日本ソフトウェア科学会会員

阿草　清滋　（正員）

1970 京大・工・電気第二卒．同大学院に進学．1974 より京都大学工学部情報工学科に勤務．同助教授を経て，1989 より，名古屋大学工学部電気第二学科教授．現在名古屋大学情報メディア教育センター教授．工博．ソフトウェア工学に関する研究に従事．特に要求分析，ソフトウェア仕様化技法，ソフトウェア開発モデル，再利用技術などに興味をもつ.