ソフトウェア工学 103−24
（1995．3．10）

ソフトウェア仕様書のための帳票作成機能について

山本隆広　上野和彦　黒木宏明
NTT

構造化分析・設計手法では、分析に必要な情報や詳細設計の概要のみを記述するだけで、関数インタフェースやデータ型など実装に必要な詳細な情報は取り扱わない。本稿では、上流工程と下流工程の作業の連続性をとるために、構造化分析手法・設計によりシステムの上流工程を設計し、次に、下流工程の情報を帳票形式に整理して開発する作業モデルを提案する。さらに、このモデルにもとづいて上流工程のCASE情報を変更した時に、下流工程の帳票を更新・矛盾チェックする方法を提案・実装する。このことにより、構造化分析・設計法で作成した上流工程のCASE情報と下流工程の帳票形式の情報の一貫性をとることができる。

Table Generate System for Software Development

Takahiro YAMAMOTO　Kazuhiko UENO　Hiroaki KUROKI
NTT

Structured Analysis and Design (SA/SD) methods define the analysis information and the abstract of design , and , they don't define the implementation information ; data type , function , etc. This paper suggests the work model which connects the analysis information with the implementation information using the design information table , the algorithm for updating the design information table in order to keep consistency when the analysis information is changed in the work model , and the system for the implementation of the algorithm. The system can keep consistency of the analysis information made by CASE system and the implementation information.

1 はじめに

ソフトウェアの開発、分析方法論として構造化分析・設計手法[8] (Structured Analysis / Design) が提唱されている。構造化分析・設計手法は、(1) 対象業務のデータの処理とデータの流れに着目することにより、データフロー図を記述する、(2) これをレビュすることにより対象業務で取り扱うデータとデータの関係を抽出する、(3) 作成したデータフロー図をもとにデータの構成を表すデータ構造図、モジュール構成を表すモジュール構造図、を作成しER図を作成する、(4) 作成したの各種図をもとに詳細設計・コーディング作業を実施する、という手順で開発する。

データフロー図・データ構造図、モジュール構造図で記述する情報は分析に必要な情報や、詳細設計の概要のみで、関数インタフェースやデータ型など実装に必要な詳細な情報は含まれていない。このため、構造化分析・設計手法であつかわれる上流工程の情報と、コーディング作業で取り扱われる下流工程の情報にギャップがあった。

これに対して Popkin [1]や Stp [1] などのCASE ツールでは、上流工程の情報をデータ辞書から抽出して、データ名一覧などの帳票形式や定型書式に印刷し、次の工程の仕様書やレビュの資料とすることができた。

しかし、これらの CASE ツールでは、データ辞書の内容から新規に帳票をつくるため、下流工程の作業において、作成した帳票に情報を追加した時に、CASE の情報を変更して、帳票を更新すると、帳票が再作成されて、追加した情報が消えてしまった。このため、上流工程で作成したダイアグラムと下流工程で使用する帳票の一貫性をとることがむずかしかった。

そこで、本稿では構造化分析手法・設計によりシステムの上流工程を設計し、次に、コーディングのために必要な関数インタフェースなどを帳票形式に整理して開発する作業モデルを提案し、このモデルに基づいて、上流工程の CASE 情報を変更した時に、下流工程の帳票を更新・矛盾チェックする方法を提案・実装する。このことにより、構造化分析・設計法で作成した上流工程の CASE 情報と下流工程の帳票形式の情報の一貫性をとることができ、帳票を介して上流の工程の情報に、コーディングに必要な情報を連続的に付加させることができるので、上流工程の作業と下流工程の作業の情報の連続性を保つことができる。

2 構造化分析・設計手法

Yordon らによって提案された構造化分析・設計手法は以下の手順で実施される。

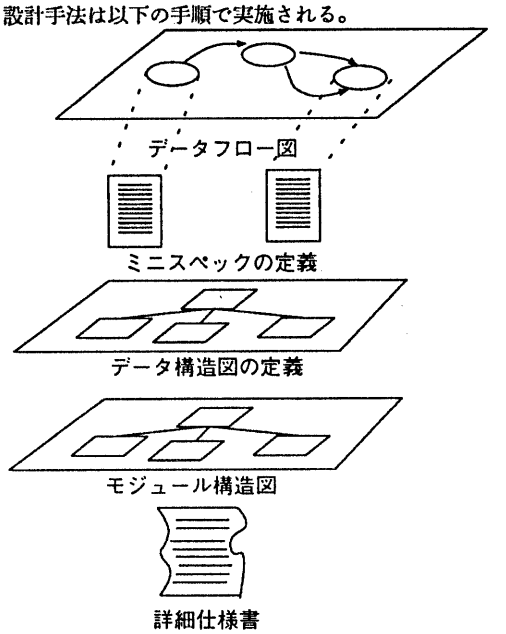

図1 構造化分析・設計手法

(1) データフロー図作成 対象業務のデータの流れ (データフロー) とデータの処理 (プロセス) に着目して、データフロー図を記述する (図1)。作成したデータフロー図のプロセスを段階的に詳細化することによって、データフロー図を上位から下位のレベルのデータフロー図を作成していく。

(2) データフロー図の整理 作成したデータフロー図の階層構造を整理し、1つの図面に6～7個のプロセスがはいるようにする。またプロセス名が具体的な動作を記述している場合は、論理的な意味を表すように変更する。

(3) ミニスペックの定義 最下層のレベルのデータフロー図のプロセスの詳細を文章でミニスペックとして記述する。

(4) データ構造図の作成 データフロー図で定義したデータの構成をデータ構造図によって記述する。

(5) モジュール構造図の作成　中央変換法などの方法によりデータフロー図から機能の中心となるプロセスを選択し、そのプロセスを上位にモジュール構成図を作成する。

(6) モジュール構造図の整理　モジュールの凝結度、実装時に必要な入出力モジュールの追加などを考慮して、作成したモジュール構造図を整理する。

(7) 詳細仕様書作成　作成したデータ構造図、モジュール構造図をもとに、モジュール内の具体的な処理とデータ型などのデータ構造を定義する。

(8) コーディング　定義した詳細仕様書に基づいてコーディングを実施する。

3　作業モデル

2章の構造化分析・設計手法では、データ構造の数、モジュール構造の数が増えると図が複雑になるため可読性が低下してしまう。このため、大局的にシステム構造の方針を検討するには、データ構造、モジュール構造図ともに、概要を記述するのみにとどめる必要があった。しかし、これでは、(6) の作業の情報と (7) の作業の情報に記述レベルでギャップができてしまう。そこで、(6) の作業と (7) の作業の継続性を円滑にするめるために、本稿では、構造化分析・設計手法から帳票形式によって詳細情報を整理・付加し、帳票に記述された詳細情報からコーディング作業をする作業モデルを提案する。

(1),(2) データフロー図の作成・整理　実際の分析対象の業務名を使用して具体的な名前でデータフロー図を作成・整理する。

(3) ミニスペックの作成　実際の分析対象の業務名を使用してミニスペックを作成 a する。

(4) データ構造図の作成　(1),(2) で作成したデータフロー図のデータ構造図を作成する。

(5) モジュール構造図の作成・整理　(1),(2) で作成したデータフロー図のモジュール構造図を作成・整理する。

(6) 分析データの抽象化　SA によって作成したデータフロー図、ミニスペック、データ構造図、モジュール構造図を再び、レビュ・整理し、使用したデータ名、プロセス名を抽象的なものにし、データフロー図の階層構造、モジュール構造、データ構造を統廃合することにより、最適なものにする。

(7) 帳票テンプレートの作成　データフロー図、データ構造図、モジュール構造図、ミニスペックからモジュール名、データ名、呼びだしモジュール名などを抽出し、詳細設計書の帳票を作成するために必要なテンプレートを作成する。

(8) 帳票作成　帳票テンプレートに従い、データ型、モジュールの引数、実際の関数など必要な項目を定義していく。

(9) 実装に必要な情報の追加　帳票テンプレートに記述されている関数に、実装に必要な入出力などの部分の関数のフィールドを追加する。

(10) 詳細設計書作成　作成した帳票をもとに詳細設計書を作成する。

(11) コーディング　詳細設計書をもとにコーディング作業を実施する。

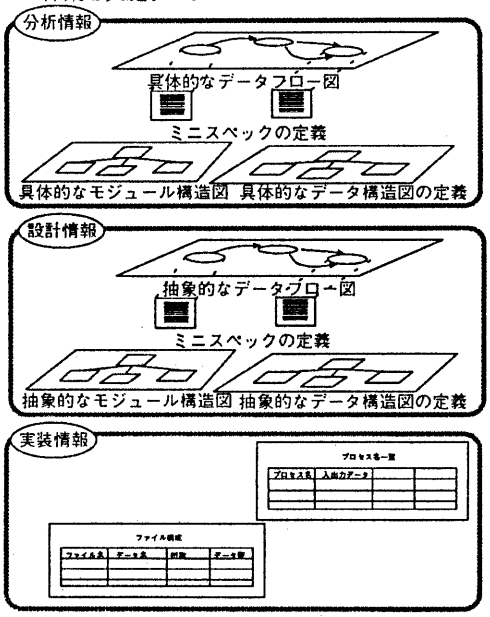

図2 作業モデル

4　データモデル

3章の作業モデルで取り扱われる情報を整理する。ここでは、帳票を作成するための情報を

「分析情報」、「詳細情報」、「実装情報」以下のように整理する。

4.1 分析情報

対象業務で取り扱われる情報を理解するために、対象業務を分析した結果を具体的に表す。ここでは、プロセス名、データ名など各ダイアグラムの要素につける名前は、実際の業務につかわれている実体の名前を使う。ここでは以下の内容を決定していく。

- データフロー図の情報
  - 対象業務から抽出した具体的な業務名をデータフロー図のプロセス名とする。
  - 対象業務で取り扱われる具体的なデータ名をデータフロー図のデータフロー名とデータストア名とする。
  - 対象業務の手続きの概要をミニスペックとして定義する。
- データ構造図の情報
  - データフロー図のデータフロー名、データストア名のデータの構成を具体的に定義する。
  - データの上限、下限、実業務上の制約条件をデータの詳細情報として定義する
- ER図の情報
  - データフロー図のデータフロー名、データストア名をエンティティにし、エンティティの具体的な関係を関係名とする。
- その他
  - システムを使用するユーザの具体的な要求条件、例えば、ターンアラウンドタイム、処理できる件数、格納できるデータ数などを分析することにより定義する。

4.2 設計情報

システムの論理的な構成を決定するために、分析情報で対象業務から抽出した情報を抽象化して定義する。

- データフロー図の情報
  - 分析情報として定義したデータフロー図を抽象化したプロセス名を新しいデータフロー図のプロセス名とする。
  - 分析情報として定義したデータフロー図を抽象化したデータ名、データストア名を新しいデータフロー図のデータ名、データストア名とする。
- データ構造図の情報
  - 抽象化したデータフロー図に合わせて、データ名、データの構成を抽象化して定義する。
  - データの具体的なデータの制約条件の他に、データの論理的なデータ型を記述する。
- ER図の情報
  - 抽象化したデータフロー図、データ構造図のデータ名に合わせてエンティティ名、関係名を抽象化する。また、抽象化したときに、同一の内容となるエンティティ、関係は統合化する。
- モジュール構造図
  - 抽象化したデータフロー図のプロセスを参考にしながら、作成システムのモジュール構造を定義する。
  - データフロー図のミニスペックの手続きを参考にしながら、モジュールの詳細を定義していく。
- その他
  - データフロー図のミニスペック、モジュール構造図を参考に各モジュールのアルゴリズムの概要を定義していく。

4.3 実装情報

分析情報、設計情報をもとにシステムを実装するために必要な情報を定義する。

- スキーマ
  - ER図を参考にスキーマを定義する。
- データ構造

- データ構造図、ER図を参考にコーディング時に定義するデータ型を定義する(C言語ならばstruct , enum , typedef など)。

- ファイル構造

  - ファイル名、ファイルの構成要素、ファイルの格納形式を定義する。

- モジュール構造

  - クライアント・サーバ方式の構成やマルチプロセスの構成でシステムを構築する時は、プロセス名、各プロセス間のプロトコルを定義する。

  - コーディング時に定義する関数の入出力インタフェース(関数名、引数、リターン値、データ型、引数の制約条件)を定義する。

5 帳票作成機能の概要

本稿では 4 章で定義したデータモデルのうち、設計情報、実装情報の情報のギャップを埋めるために、レビュ用の帳票を定義し、その帳票の作成支援機能を提案する。

設計情報では、データ名、プロセス名などの実際の業務を抽象化することにより整理したものを記述する。これに対して、実装情報では、実際のデータ型や関数名などの実装に依存した項目を定義する。ここでは、この設計情報の各項目に対応して実装情報の項目を定義した対応表をレビュ用の帳票とすることにより、設計情報と実装情報の対応を明確にする。

従来の 上流CASE ツールでも、分析・設計作業の情報と下流工程の情報の対応を明確にするため、データ辞書から定型的な帳票を生成することができた。しかし、上流工程の情報から一方的に帳票を生成するのみで、上流工程の情報を変更した時に、帳票を更新したり、下流工程の情報とマージすることが難しかった。本稿では、前章までで定義した作業モデルとデータモデルに基づき、帳票の更新、マージ機能を改善することによりCASEで作成した上流工程の情報と下流工程の情報の一貫性をとる方法を述べる。

5.1 帳票作成機能のスキーマ

帳票の構成要素として以下のものがあげられる。

- データフロー図の機能(プロセス)と入出力データの関係
- データフロー図の機能(プロセス)とデータストアの関係
- データ構造図のデータ名、データの構成要素、データ型の関係、制約条件
- データ構造図のファイル名、ファイルの構成要素
- モジュール構造図のモジュール名、関数名、モジュールの引数、モジュールの引数、リターン値のデータ型

5.2 帳票作成機能の支援機能

5.2.1 帳票データの特徴

4 章で定義したデータモデルおよび、5.1 章のスキーマには以下の特徴がある。

- 上流CASE ツールなどで設計情報がつくられることが多いが、多くの場合、ユーザの判断で実証情報をつけ加えなければならなく、完全に自動的な処理はできない。
- 設計情報に詳細な情報がつけ加えることにより実装情報が定義されている。
- 設計情報の作成および実装情報の作成は複数の作業者で行なわれることがある。

本稿では、上記のスキーマの帳票をCASEの情報から作成する以下の機能を提案する。

帳票データ抽出機能 上流CASE ツールを使用して分析・定義された設計情報から帳票作成に必要がデータを抽出する機能。

帳票データマージ機能 各ツールにより作成された複数の帳票データを一つにまとめる。

帳票データ流しこみ機能 作成した帳票データを通常のドキュメントの形に清書するために、DTP やワードプロセッサの文書に流し込む。

帳票更新機能 帳票データが更新されたときに、連動して DTP やワードプロセッサに流し込まれて清書された文書も更新する。

帳票データ矛盾チェック機能 マージした帳票データや更新した帳票データの定義に矛盾がないかチェックする。

5.2.2 帳票データのマージ・矛盾チェックの方法

ここでは、5.2.1を参考にしながら、帳票データのマージ方法、矛盾チェックについて述べる。以下の方針でアルゴリズムを検討した。

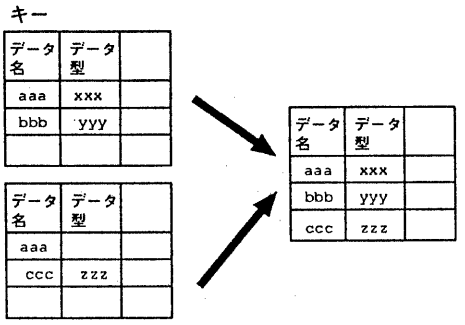

図3 帳票データのマージ・矛盾チェックの方法

(A) 矛盾の検出のために、帳票のある列をキーとして定義する。

(B) 唯一のキー列があり、列内容が空白のものはそのままマージする。

(C) 同じキー列で かつ 異なった列内容があった時には、該当する行をすべて流し込み、後でユーザがチェックできるように、該当する行にハッチングをかける。

(D) 帳票をマージした時に、同じキー列で かつ 同じ列内容 (キー列以外の列) があった場合、一つの行にまとめる。

3 章で提案した作業モデルでは、実装に必要な情報を設計情報に付け加えることにより実装情報を定義する。そこで、最初に設計情報でテンプレートを作成しておき、実装情報を定義する部分を空欄にしておく。次に、設計者が空欄に必要な情報を記入することにより実装情報を定義する。このとき矛盾チェックのアルゴリズムは、追加された情報を優先してとりあつかう (B)。

また、複数人で実装情報を定義した時に、キー列が同じで、かつ、キー列以外の欄が異なる時は、その行は矛盾しているとして、ハッチングをかけることにより、後で設計者の判断で修正できるようにしておく (C)。

複数人で実装情報を定義した時に、同じキー列であり、かつ、同じ列の内容のときは、同じ情報であるとして、一つの行として縮退させる (D)。

5.3 帳票の更新機能

上流 CASE ツールなどで、上流工程のデータを変更したときに、DTP やワードプロセッサの帳票の内容も上流のデータに合わせて更新する必要がある。以下の手順で更新を行なう。

(A) DTP、ワードプロセッサのタイムスタンプと、帳票データのタイムスタンプを比較する。

(B) 帳票データのタイムスタンプが、DTP、ワードプロセッサのタイムスタンプより新しければ、帳票データが更新されているとして、DTP、ワードプロセッサの帳票の内容と帳票データをマージする。

(C) マージしたデータに 5.2.2 章の矛盾チェックをかけることにより、余分な行の縮退、矛盾行のチェックをする。

(D) チェックしたデータを、DTP、ワードプロセッサの元の帳票内容と入れ換える。

帳票データのマージと同様に、帳票の更新時のときにも 5.2.2 章の矛盾チェックをする。このことにより、帳票データを更新したときに、DTP、ワードプロセッサの帳票に内容があり、かつ、帳票データにデータがなければ、DTP、ワードプロセッサ側のデータを優先させる。

また、更新した帳票データと、DTP・ワードプロセッサの帳票の内容で、キー列が同じで、それ以外の列の内容が違う行があれば、帳票データと DTP・ワードプロセッサの帳票の内容に矛盾があるとして、ハッチングをかけて修正が必要な部分を示す。

5.4 帳票作成機能の実装

本稿で提案した帳票作成機能を実装した。実装した環境は、SparcStation 10 で、OS は、Solaris2.3J である。実装した部分は以下のものからなる。

上流 CASE 構造化分析・設計手法を支援する CASE ツール。今回は、SoftDA を使用した。

データ辞書アクセス部 CASE ツールのデータ辞書にアクセスし、帳票に必要な情報をよみこみ、帳票データとして出力する。今回、帳票データとして出力する形式は、市販のスプレッドシートなどで、つかわれている CSV 形式を採用した。

帳票データマージ部 CSV 形式によりフォーマットされた複数の帳票データをマージする。

矛盾チェック部 / 帳票データ流しこみ部 マージされた帳票データを 5.2.2 章のアルゴリズムにもとづき、矛盾をチェックする。そして、矛盾箇所にハッチング処理をして、ワードプロセッサの文書に流し込む。

帳票更新部 文書のタイムスタンプと帳票データのタイムスタンプを比較し、5.3 章のアルゴリズムに基づいて更新の必要がある時は、帳票データマージ部 を起動して、帳票を更新する。

ユーザインタフェース部 帳票ファイル類の指定、作成する帳票のスキーマの選択をするユーザインタフェース部。GUI により、矛盾チェックのためのキー列の指定、ファイルの指定を行なうことができる。

スプレッドシート起動部 作成する帳票に項目番号やソートなどの処理をするときに、帳票の内容をスプレッドシートに読み込ませ、項目番号やソートなどの処理をスプレッドシートで施した後、再び帳票に戻す部分。

DTP 今回は、市販の UNIX 上の DTP ツールを使用した。DTP で帳票データを流し込む帳票のテンプレートを作成したり、実装情報を付け加えるときに、帳票の空欄に必要な項目を入力する。

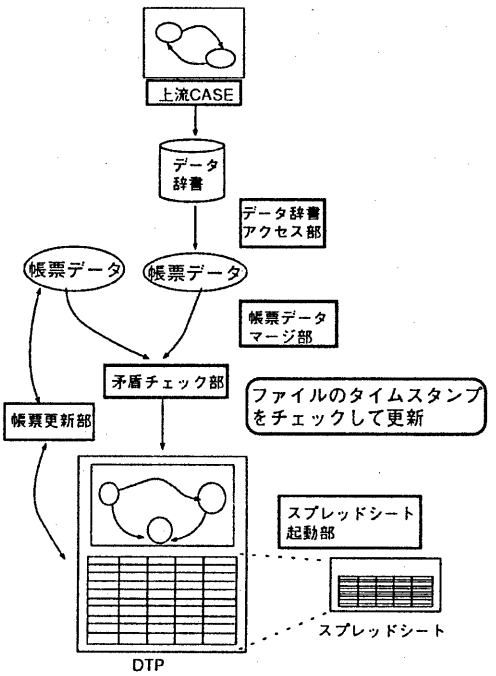

図4 帳票作成支援機能

6 関連研究

Information Engineering を支援している統合化 CASE ツールである、IEF,IEW では、分析工程で作成された各種図から自動生成のために必要な詳細情報を定義するために、マトリックスを作成することができる。しかし、これらのツールは自動化を目的としてマトリックスを定義するため、マトリックスの書式が固定的で、各プロジェクトにあわせ書式で出力することができないため、各プロジェクトにあわせたレビュに使用するのは難しい。本稿で提案した実装方法では、DTP の流し込み部およびデータ辞書アクセス部の定義を変更することで、生成する帳票の書式をカスタマイズすることができる。

構造化分析・設計手法をサポートする上流CASEであるStP、Popkinでは、レポートジェネレータにより、マクロ命令のテンプレートにしたがい、データ辞書の内容から詳細仕様のテンプレートやデータ名、プロセス名の一覧を帳票として作成することができる。しかし、上流工程製造物であるダイアグラムなどを変更した場合、帳票を再出力しなければならず、設計工程で帳票の空欄に追記した情報が消えてしまう。本稿で提案した矛盾チェックアルゴリズムでは、

- 「設計情報」へ実装に必要な項目が付加されることにより「実装情報」が定義されていること。

に着目して、後から追記した情報を優先する形で帳票の更新を行なっているため、上流工程製造物であるダイアグラムなどを変更した場合にも、既存の帳票の情報を半自動で保存することができる。

7 まとめ

本稿では、構造化分析・設計手法を詳細化するときに、帳票形式のテンプレートで整理し、この帳票を詳細化・レビュすることにより設計作業をすすめる作業モデル、および支援機能について述べた。

また、本稿で示した矛盾チェック・マージアルゴリズムにより、上流工程のダイアグラムを変更しても、下流工程の情報を優先してマージすることにより、下流工程の情報を保存して帳票を更新することができる。

今後は、本稿で提案した帳票支援機能の有効性を確認するために、実際の設計・分析作業に適用していくことが考えられる。

参考文献

(1) 原田 実:CASE のすべて, オーム社,1991

(2) 黒木、山本: 構造化分析手法とデータベース設計技法の統合の試作, KBSE93-33, 信学技報,1994-01

(3) 原田、大平: 出力様式から形式的要求仕様を生成する要求分析システム GRACE, 信学論文誌 D-1,1994-09

(4) 西村, 本位田: 複合ビューポイントに基づく仕様化プロセスの分析, 情処論文誌,1993-05

(5) 山本、黒木: ソフトウェア仕様書のための帳票作成機能について, 春季信学全国大会,1994

(6) 藤井、他: ソフトウェア開発支援システム SDSS における CASE 統合化, 情処論文誌,1995-01

(7) DeMarco.T: 構造化分析とシステム仕様, 日経 BP 社,1986

(8) Yourdon.E: Modern Structured Analysis,Prentice-Hall.Inc.,1989