ソフトウェア工学 81－2
（1991. 10. 4）

モジュール構造設計支援機能の提案

西永 誠司　山本 修一郎　磯田 定宏

NTT ソフトウェア研究所

あらまし　構造化設計法に基づくモジュール構造設計を行う上での問題点の一つに、データフロー図を基に設計したモジュール構造はソースコードと完全には対応していないため、コード化には必ずしも適さないことがあげられる。

本稿では、データフロー図からモジュール構造を設計する作業をモデル化し、コード化に適したモジュール構造を生成するアルゴリズムを提案する。また、実際にモジュール構造設計にアルゴリズムを適用し、その有効性の評価を行う。さらに、アルゴリズムに基づいた、モジュール構造設計支援システムの実現法についての検討結果を述べる。

A Design Method for Software Module Structure

Seiji Nishinaga　Shuichiro Yamamoto　Sadahiro Isoda

NTT Software Laboratories

Abstract　The logical module structure which is mechanically transformed from data flow diagram doesn't always correspond with the program code. Designer should manually make up for the gap between logical and code level module structures. It is not obvious how to design code level module structure from logical module structures.

At first, we present the model of module design process and the design method for module structure based on this model. Next, we estimate the effectiveness of the method. Finally, we propose the module design support system based on the model.

1 はじめに

モジュール構造設計を行う上での問題点の一つに、データフロー図を基に設計したモジュール構造は、ソースコードと完全には対応していないことがあげられる。

これは、ソフトウェアの要求仕様を表したデータフロー図を基に、機械的に作成したモジュール構造は、以下の理由でコード化に適しているとはいえないからである。(図1)

- 十分に詳細化されていないモジュールが存在する
- 実質的な処理を持たない、冗長なモジュールが存在する
- コード化する際に必要な入出力データが不足している
- 冗長な入出力データが存在する

本稿では、この問題を解決するための一方策として、データフロー図からモジュール構造を設計する手順をモデル化し、ソースコード作成に適したモジュール構造を生成するアルゴリズムを提案する。そして、このアルゴリズムを基にした**「モジュール構造設計支援システム」**の実現法についての検討結果を述べる。

2 モジュール構造設計作業モデル

データフロー図からモジュール構造を設計する作業を、以下のようにモデル化した。(図2)

Step.1 変換

要求仕様を表したデータフロー図を、変換分析法やトランザクション分析法などの手法を用いて機械的にモジュール構造に変換する。

Step.2 整形

Step.1の変換で得られたモジュール構造に対して、モジュールやデータの追加, 統合, 共通化を行い、最適化する。

Step.3 一貫性検証

Step.2で整形したモジュール構造が、もとの要求仕様を満たしているかどうか、データフロー図との一貫性を検証する。

以下では、モジュール構造設計作業モデルの各ステップの作業の内容について述べる。

2.1 Step.1 変換

まず、第1ステップとして、データフロー図上の情報だけを用いて、モジュール構造を作成する。

例えば、変換分析法やトランザクション分析法では、データフローの形状を基に、データフロー図を入力部、出力部、変換部に分ける。そして、各部ごとに一定のルールに従ってプロセスとモジュールを対応付けることによってモジュール構造に変換する。

2.2 Step.2 整形

データフロー図の情報だけを用いて変換したモジュール構造をそのままコード化することは容易ではない。このようにして得られたモジュール構造は、

- 十分に詳細化されていないモジュールが存在する
- 実質的な処理を持たない冗長なモジュールが存在する
- コード化する際に必要な入出力データが不足している
- 冗長な入出力データが存在する

などの理由でコード化に適しているとはいえないからである。

このため、以下の整形を実施することにより、変換したモジュール構造をコード化に適した形に整形する。

- 初期化、終了処理、例外処理などのモジュールを追加することによる**「モジュールの詳細化」**
- 実質的な処理を持たない冗長なモジュールの削除や、類似するモジュールの共通化による**「モジュールの統合」**
- 制御情報, エラー情報などの入出力データの付加による**「入出力データの詳細化」**
- 複数のデータを構造を持ったデータにまとめる**「入出力データの統合」**

2.3 Step.3 一貫性検証

Step.1で得られるモジュール構造は、データフロー図を変換したものであり、対応は取れている。Step.2でコーディングに適するように実施した整形で、データフロー図、すなわち要求仕様との一貫性が維持できているかどうかチェックする必要がある。

本稿では、以下の2条件がともに満足されるとき、一貫性が満たされていると判断する。

(1) **プロセスとモジュールの対応条件**

データフロー図上のすべてのプロセスがモジュールで実現されていること

(2) **データフローと入出力データの対応条件**

データフロー図上のすべてのデータフローがモジュール間の入出力データで実現されていること

3 モジュール構造生成アルゴリズム

3.1 モジュール構造整形作業の分析

モジュール構造設計作業モデルでは、第1ステップとして、データフロー図から機械的に生成したモジュール構造を、ソースコード生成に適した構造に整形する。このとき、どのような整形作業を行う必要があるかを洗い出すため、人間が行うモジュール構造設計作業を分析した。

具体的には、実在のプログラムについて、データフロー図と、それを基に人間が設計したモジュール構造との対応関係の分析を実施した。

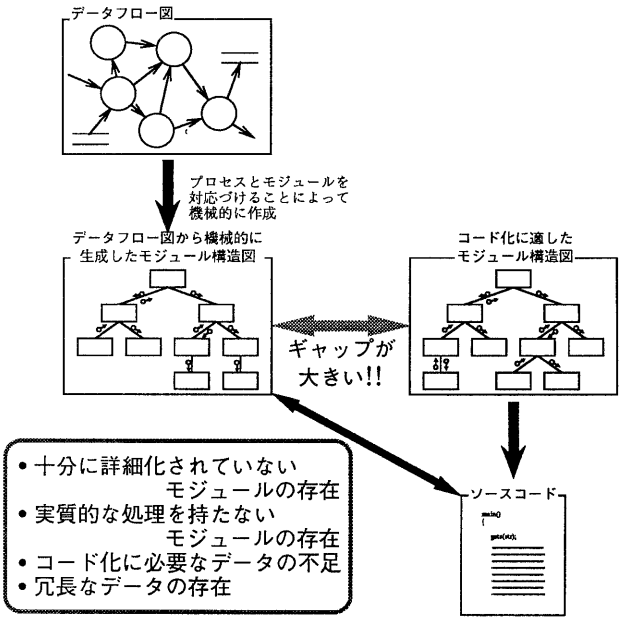


図 1: モジュール構造設計上の問題点

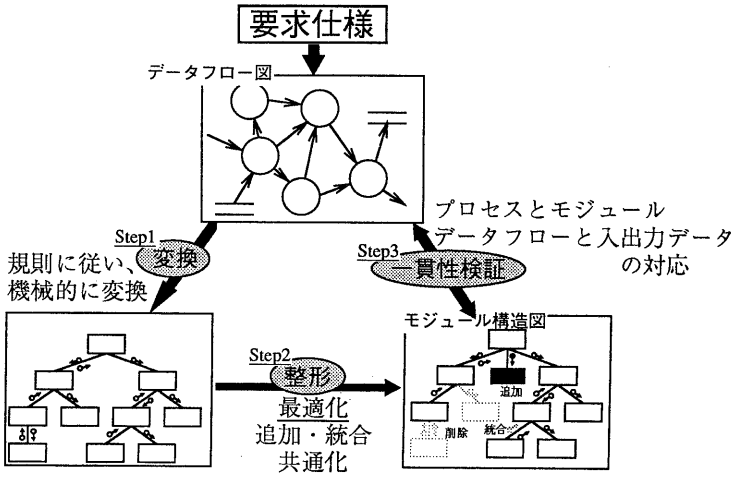


図 2: モジュール構造設計作業モデル

3.1.1 モジュールの分析

モジュール構造図上のモジュールとデータフロー図上のプロセスとを比較したところ、モジュールには、

- **プロセスと対応するモジュール**
- **プロセスと対応しないモジュール**

があり、プロセスに対応しないモジュールは以下の4種類に分類できた。

1. **初期化、終了処理**
   初期値の設定、一時ファイルの削除など
2. **例外処理**
   エラー処理など
3. **ファイル処理**
   ファイルのオープン, クローズ, 読み込み, 書き込みなど
4. **付加処理**
   ある処理に付随して必ず行わなければならない処理

3.1.2 入出力データの分析

モジュール構造図上のモジュール間の入出力データとデータフロー図上のデータフローとの対応を調べたところ、入出力データには

- **データフローと対応する入出力データ**
- **データフローと対応しない入出力データ**

があり、データフローと対応しないデータは以下の3種に分類できた。

1. **ファイル処理データ**
   ファイルポインタ, ファイル終端記号など、ファイル処理を行なう場合に必要なデータ
2. **制御フラグ**
   エラーフラグ, 処理の条件分岐の振り分け用フラグなど制御用フラグ
3. **定型データ**
   真偽値, 入出力バッファのサイズなど、プログラムで一般的に使用される定型的データ

3.1.3 分析結果

プロセスと対応しないモジュールやデータフローと対応しない入出力データは、データフロー図上では表現されていないが、コーディングを行う上で必要となる情報であるといえる。これらの情報は、モジュール構造設計時に人間の判断で付け加えなければならない。

3.2 モジュール構造の評価軸

本検討の目的は、コード生成に適したモジュール構造を作成することである。そのため、**コード生成に適したモジュール構造**とはどのようなものかを定義しておく必要がある。

コード化に適したモジュール構造の必要条件として、「**モジュール構造図の構成要素とプログラムの構成要素との対応づけが可能である**」ことがあげられる。すなわち、

- モジュールとプログラムの構成単位（Cであれば関数）が対応づけられる
- モジュール間のデータの流れと、プログラム上で実現されているデータの受渡し（パラメータ、変数など）が対応づけられる

ことが必要となる。

3.3 モジュール構造生成アルゴリズム

3.1の分析結果に基づき、データフロー図からコード化に適したモジュール構造を生成するアルゴリズムを以下に示す。(モジュール構造設計作業モデルのStep.1 変換,Step.2 整形に対応する。)

[Step.1] モジュール構造への変換

データフロー図から得られる情報（プロセスの階層関係, データフローの方向 など）を基に、機械的にモジュール構造へ変換する。

[Step.2] モジュール，入出力データの追加

追加規則（表1）に従い、コーディング時に必要となるモジュール，入出力データを追加する。追加規則は、ライブラリ化しておく。

[Step.3] モジュール構造の再構築

以下の作業を行い、モジュール構造を再構築する。

ただし、作業に先立って、「実施する」/「実施しない」の判断を下す必要がある。

(1) モジュールの統合（図3）

- ◆ 同レベルの類似モジュールを統合する
  - (i) 類似モジュールを一つのモジュールにまとめる
  - (ii) データの統合を行う
  - (iii) 処理の振り分け用制御情報を追加する
- ◆ 一レベル上位のモジュールと統合する
  - (i) モジュールとして独立させる必要性のない処理を上位モジュールに吸収する
  - (ii) 入出力データの統合を行う

(2) モジュール構造の再構成（図4）

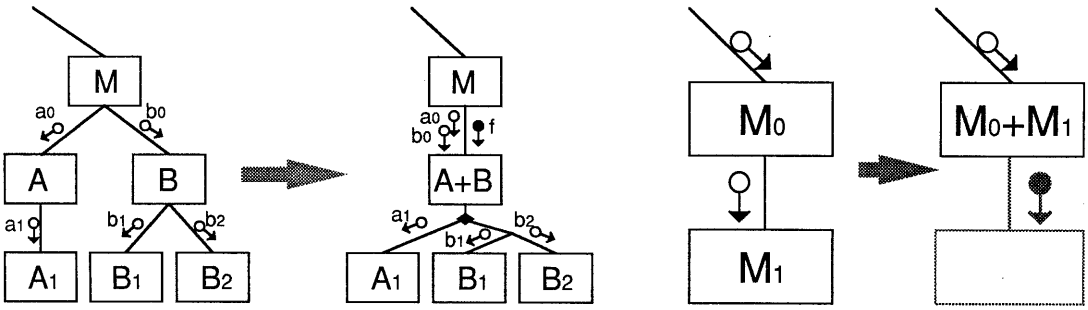
- ◆ 初期化／終了処理モジュールなど、類似の処理を行うモジュールを一か所にまとめる
  - (i) 類似モジュールを一か所に移動する
  - (ii) データ入出力の整合をとる

3.4 アルゴリズムの適用と評価

SA/SD手法で設計/製造されたシステムのデータフロー図に対して、モジュール構造生成アルゴリズムを適用し、モジュール構造図を作成する。このモジュール構造を検証することにより、本アルゴリズムの有効性を評価する。

表 1: データ／モジュールの追加規則の例

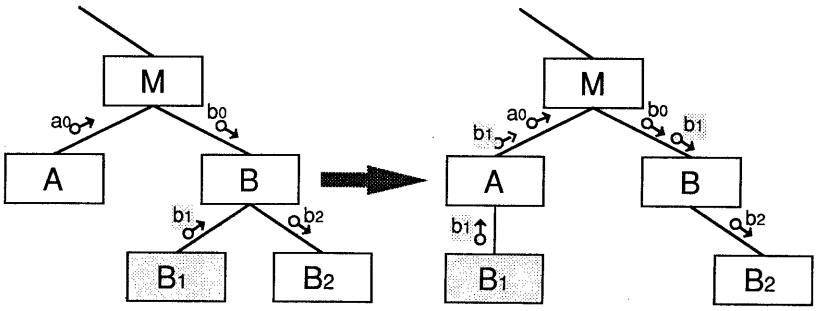
	モジュール	データ
プログラムで一般に必要とされる	・開始処理 ・終了処理 ・例外処理	・エラーフラグ ・制御フラグ
ファイルへのアクセスがある場合に必要とされる	・ファイルのオープン ・ファイルのクローズ ・ファイルの読み込み ・ファイルへの書き込み	・ファイルポインタ ・ファイル終端記号
その他、条件により必要とされる	・プロファイル情報の取得 ・環境変数の取得	・プロファイル情報 ・環境変数情報



1.モジュールAとBを統合する
2.処理の振り分け用のフラグfを追加する

同レベルの類似モジュールの統合　　一レベル上位のモジュールとの統合

図 3: モジュールの統合



1.モジュールB1をモジュールBの下からモジュールAの下に移動する
2.モジュールB1からモジュールBへのデータb1の経路を作成する

図 4: モジュールの移動

3.4.1 適用例

- 適用対象

  メッセージ転送監視対象センタ登録 / 照会コマンド

- 機能概要

  メッセージ転送監視処理システムのサブシステム。

  監視対象センタ情報を格納したファイルへの登録 / 削除 / 照会を行う。

- データフロー図規模
  1 階層, プロセス数 4

- コード記述言語・規模
  SYSL, 約 800step

- アルゴリズム適用時に実施した最適化処理

  1. 同一のデータストアへの入出力処理を共通化する
  2. 類似処理 (プロセス名称で判断) を共通化する
  3. 構造の変更 (登録 / 照会 / 削除の各処理を条件による振り分け処理とする)

- 適用結果

  アルゴリズムを適用して作成したモジュール構造とコード化された段階でのモジュール構造の比較を行う。

  各モジュール構造を図 5に、モジュールの対応を表 2に示す。

3.4.2 評価

モジュール構造図とコードの構造の比較結果より、アルゴリズムに基づいて生成した構造は、コードで実現されている機能を全て含んでおり、コード化段階のモジュール構造を詳細化した形になっているといえる。

ただし、コード化された段階での構造に比べてモジュール数は多くなっている。この理由は、

1. システムの処理内容が比較的単純で、ファイル (データストア) への入出力が主となっている
2. コーディング段階では、これらの入出力処理を独立なモジュールとするのではなく 1 モジュールで実現している

ためである。

アルゴリズムを適用して作成されたモジュール構造を基にコーディング作業に移行することは容易であると考えられる。

ただし、アルゴリズム適用時の整形段階で、以下の点を考慮する必要がある。

- 処理効率を考慮したモジュール構成の変更
- データの受渡し方法など、コーディング時の実現法を反映した構造の変更

4 モジュール構造生成支援システムの提案

4.1 モジュール構造整形作業における判断

モジュール構造整形作業時に、各種の判断を下す必要がある。この判断は、

(1) インタラクティブに人間が指定する

(2) 推論を行い、システムが行う

の 2 通りの方式が考えられる。

(2) の方式では、システムが判断を下すために、推論のためのしかけ (知識ベースなど) が必要となる。また、判断の基準とするものを明確にしなければならない。

4.2 モジュール構造設計支援システムの実現法

モジュール構造設計支援システムを実現するうえで、以下の 3 段階のアプローチを考える。

◆ **第一段階**

設計作業は、**人間**が行う。

システムは作業支援環境を提供する。

◆ **第二段階**

設計作業のうち、機械的に行える作業については、**システム**が行う。

判断が必要な作業については、インタラクティブに**人間**が判断を下す。

◆ **第三段階**

判断が必要な作業についても、**システム**が推論を実施して行う。

人間は微調整程度の作業を行う。

4.3 モジュール構造設計支援システムの機能

2. で提案したモジュール構造設計作業モデルに基づいてモジュール構造の設計を行う場合、以下の点に留意する必要がある。

- 整形を行うとき、その変更にともなう影響範囲を明確にしなければならない
- 一貫性を検証するため、データフロー図とモジュール構造図といった形式の異なるもの同士の比較を行わなければならない

本稿では、設計情報リポジトリの機能を利用してモジュール構造設計作業を支援する方法を検討する。設計情報リポジトリでは、ソフトウェアの各種の情報を互いに関連した形で記録している。要求仕様、プログラム構成、モジュール仕様などソフトウェアを構成する各種の情報を一元管理することにより、設計情報間の無矛盾性のチェックや、修正を加えた場合の影響範囲の特定などを行なうことが可能となる。 [3]

モジュール構造設計支援システムで実現する機能の一覧を表 3に示す。

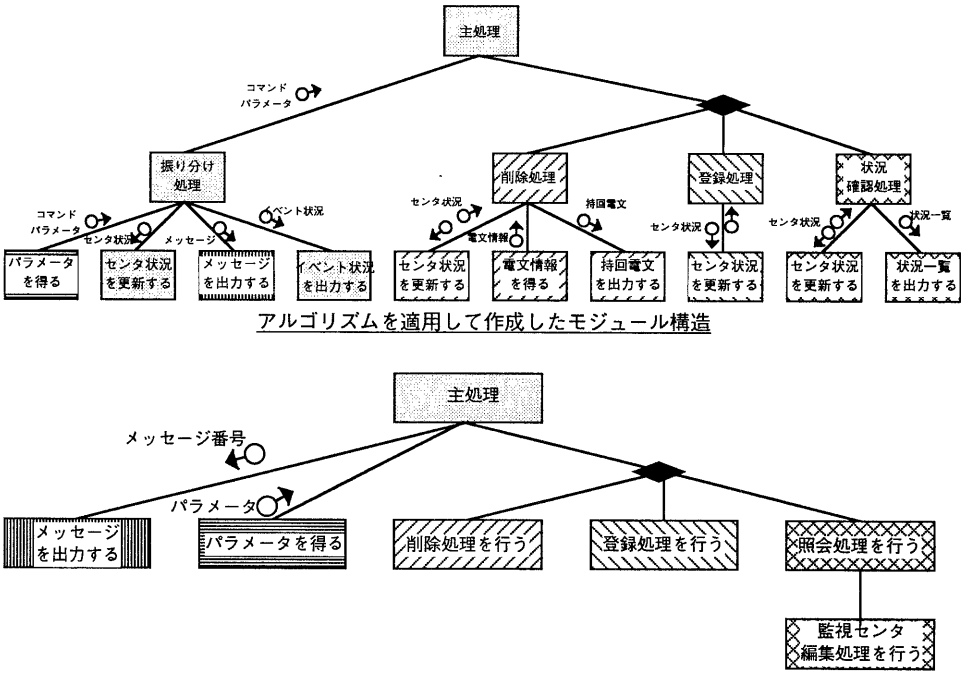


図 5: メッセージ転送監視対象センタ登録 / 照会コマンド: モジュール構造図

表 2: 適用例 1: モジュールの対応表

アルゴリズムを適用して作成したモジュール構造	コード化された段階でのモジュール構造	図5でのマーク
主処理 振り分け処理 イベント状況を出力する センタ状況を更新する*	主処理	
削除処理 電文情報を得る 持回電文情報を出力する センタ状況を更新する*	削除処理を行う	
登録処理 センタ状況を更新する*	登録処理を行う	
状況確認処理 状況一覧を出力する センタ状況を更新する	照会処理を行う 監視センタ編集処理を行う	
パラメーターを得る	パラメーターを得る	
メッセージを出力する	メッセージを出力する	

*:共通モジュール

表 3: モジュール構造設計作業モデルにおける支援機能の一覧

項番	機能名称	機能概要
1	対応関係情報獲得機能	プロセスとモジュール，データフローとモジュールコール関係の 対応付けを支援する
2	設計情報管理機能	設計情報リポジトリへの設計情報の登録・修正・削除を行った場合の データの一元管理を行う
3	設計情報検索機能	設計情報間の関係をたどることによって 検証時の検索作業を効率化する
4	一貫性検証機能	プロセスとモジュール，データフローとデータ (パラメタ) の入出力関係 の対応関係をチェックし、要求仕様との一貫性を検証する
5	DFD → SC 変換機能	データフローダイアグラムを 機械的にストラクチャチャートに変換する
6	モジュール構造整形機能	モジュール構造の整形作業を支援する

◆ 対応関係獲得機能

モジュール構造設計時に、プロセスとモジュール、データフローとモジュールコール関係の対応を設定するため、モジュール構造設計用のエディタから、設計情報リポジトリとのインタフェース画面を起動し、作業の支援を行う。

◆ 一貫性検証機能

設計情報リポジトリで管理する情報を基に、ストラクチャチャートとデータフローダイアグラムの一貫性を検証する。

◆ モジュール構造整形機能

モジュール構造整形時に判断が必要となる作業について、以下のような支援を行う。

- 類似の同レベルのモジュールを統合するかどうか？
  - 類似モジュールを名前を基に検索し、提示する。
  - (統合する場合) モジュールの統合によるデータの統合, 削除, 振り分け用制御情報の追加を行う。
- 一レベル上位のモジュールと統合するかどうか？
  - 候補となるモジュールを提示する。
  - (統合する場合) モジュールの統合によるデータの統合, 削除を行う。
- 複数のデータを統合するかどうか？
  - 統合の候補となるデータを検索し、提示する。
- 初期化／終了処理モジュールをまとめるかどうか？
  - 候補となるモジュールを検索し、提示する。
  - (移動する場合) 移動によるデータの入出力関係の再構成を行う。

4.2 システムの実現法の各段階において、モジュール構造設計作業モデルに適用する支援機能を表 4 に示す。

表 4: モジュール構造設計作業モデルへの適用機能

モジュール構造設計作業モデル	支援システムの実現段階		
	第一段階	第二段階	第三段階
Step.1 変換	1	1,5	1,5
Step.2 整形	2,3	2,3,6	2,3,6
Step.3 一貫性検証	4	4	4

ここで、i($1 \leq i \leq 6$) は、表 3で示したモジュール構造設計支援機能の項番 i の機能であることを示す。

5 まとめと今後の課題

本資料では、モジュール構造設計作業をモデル化し、データフロー図からソースコード作成に適したモジュール構造を生成するためのアルゴリズムを提案した。このアルゴリズムを適用することにより、簡単なシステムに対しては、コード化が可能なレベルのモジュール構造を生成できる見通しが得られた。

今後は、以下の検討を進める予定である。

◆ モジュール構造生成アルゴリズムの評価と見直し

- モジュール構造の定量的な評価尺度
- 複雑なシステムへの適用と評価
- データ受け渡しなど、コード上での実現方法を反映させたモジュール構造の再構成法

◆ モジュール構造生成支援システムの試作および評価

- 一貫性検証方式のモデル化
- 推論方式の確立

参考文献

[1] Meilir Page-Jones, "The practical guide to structured system design", Prentice-Hall Inc., 1980.

[2] J J.-P.Tsai and J.C.Ridge, "Intelligent Support for SpecificationsTransformation", IEEE Software, pp.28-35, November, 1988.

[3] 岡, 山本, "設計情報リポジトリを用いた改造支援方法", 第 41 回情報処理学会全国大会,1990.