ソフトウェア工学　83－3
（１９９２．　２．　６）

統合化CASEシステムSoftDAの機能とその評価

黒木 宏明　磯田 定宏

NTTソフトウェア研究所

ソフトウェアの品質と生産性を向上するためCASEツールに大きな期待がかけられているが、まだ十分にその期待に応えているとは言えない状況にある。この理由の一つは、各工程を支援するツールはあるが上流工程と下流工程が連動していないことにある。この問題点を解決するためには、各工程で作成する設計情報をリポジトリに一元管理し、この情報に基づき上流工程の設計情報から作成可能な下流工程の設計情報を自動的に生成して入力作業を低減する機能や、設計とコードの修正を自動化する機能などを提供する必要がある。これにより、頻繁に起こる同一情報の入力作業、関連する情報を検索する作業、設計ドキュメントおよびコードの一貫性を保って修正する作業などの従来は人間が行っていた事務的かつ機械的な作業を軽減できる。この結果、人間は知的かつ生産的な作業にその能力を集中することができるようになる。

本稿では、この方針に基づいて開発した統合化CASEシステムSoftDAの自動生成機能を中心とする特徴的な機能およびその有効性について報告する。

Functionality of Integrated CASE SoftDA and its Evaluation

Hiroaki Kuroki　Sadahiro Isoda

Software Laboratories

Nippon Telegraph and Telephone Corporation

There is a large anticipation that the Computer-Aided Software Engineering or CASE will contribute a lot to the improvement of quality and productivity of software development. It, however, has not come up to our expectations. One of the reasons is that each of the CASE tools support only a specific phase of software development and they are not well integrated. In order to solve this problem, such facilities are necessary that can

a)manage the miscellaneous design information in a CASE repository,

b)generate a design document for a lower phase from upper phase design information, and

c)automate modifications to documents and program code.

This will reduce the clerical and mechanical tasks that have been performed by human engineers, such as repeated entries of the same design information, inquiries for the information relevant to the object in question, updates to documetns and program code with their consistency preserved. This paper describes the characteristic features of the indegrated CASE system SoftDA, focusing on its automatic generation facilities. Its evaluation results are also presented.

1. はじめに

ソフトウェアの品質と生産性を向上するためCASEツールに大きな期待がかけられているが、まだ十分にその期待に応えているとは言えない状況にある。この理由の一つは、各工程を支援するツールはあるが上流工程と下流工程が連動していないことにある。この問題点を解決するためには、各工程で作成する設計情報をリポジトリに一元管理し、この情報に基づき上流工程の設計情報から作成可能な下流工程の設計情報を自動的に生成して入力作業を低減する機能や、設計とコードの修正を自動化する機能などを提供する必要がある。これにより、頻繁に起こる同一情報の入力作業、関連する情報を検索する作業、設計ドキュメントおよびコードの一貫性を保って修正する作業などの従来は人間が行っていた事務的かつ機械的な作業を軽減できる。この結果、人間は知的かつ生産的な作業にその能力を集中することができるようになる。

本稿では、この方針に基づいて開発した統合化CASEシステムSoftDAの自動生成機能を中心とする特徴的な機能およびその有効性について報告する。

2. システム構成

統合化CASEシステムSoftDA(Software Design Automation)は以下の5つから構成される。構成を図1に示す。

(1)構造化分析支援ツール

構造化分析手法に基づいて機能やデータの階層的な構造や論理的な対応関係を記述する各種ダイアグラムエディタを提供する。編集可能なダイアグラムを以下に示す。

1)データフロー図
2)データ構造図
3)エンティティリレーションシップ図
4)モジュール関連図
5)状態遷移図
6)状態遷移表

また、日本語要求仕様文からのデータフロー図およびデータ構造図生成、データフロー図からのモジュール関連図生成が行える。

(2)構造化設計支援ツール

構造化設計手法に基づいて実現するシステムのプログラム構造を記述するモジュール関連図エディタを提供する。

プログラムのモジュール関連を図形で表現するので、モジュールの階層構造や呼び出し関係、データの参照関係を容易に参照・編集することができる。また、モジュール関連図からモジュール仕様書のアウトラインを自動生成することができる。

(3)詳細設計支援ツール

モジュール内の処理概要を日本語疑似コーディングで段階的に詳細化し、最終的にはコードを自動生成する機能を備えた設計言語エディタを提供する。

モジュール内の処理概要は、日本語疑似コーディングされたシステムライブラリ等のパターンを選択し、その可変部分を順次埋めることにより詳細化を行える。

日本語疑似コーディング中の日本語データ名は分野知識ライブラリ中の用語対応表を用いてニーモニックに置換する。

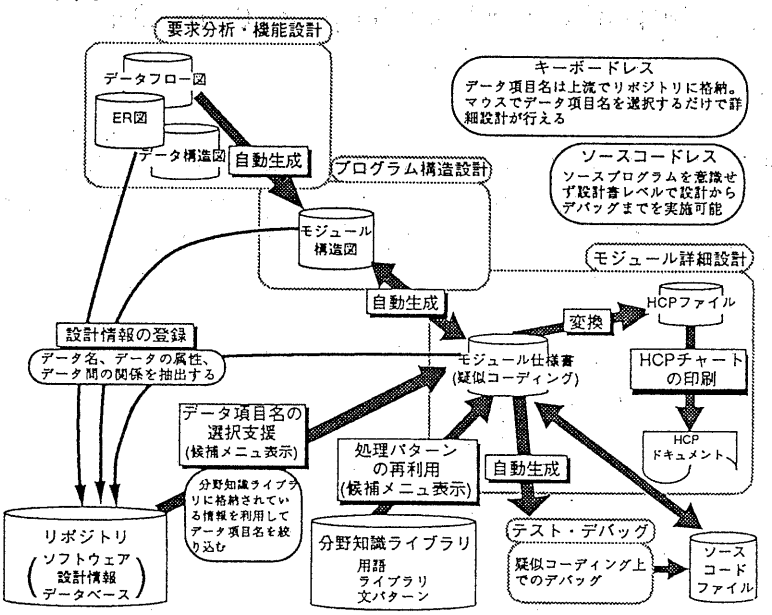


図1　統合化SoftDAの機能

(4)テスト・デバッグツール

プログラムの実行状況を視覚的に確認しながらデバッグする機能と試験の網羅率を計測する機能により、ソフトウェアの品質を向上させることができる。

疑似コーディングされた処理概要と生成コードとの対応がとられているため、処理概要レベルでデバッグすることができる。このため、ソースプログラムを必要とせずにソフトウェア開発が行える。

(5)リポジトリ

統合化システムでは、各工程で作成される設計情報を対応づけてリポジトリ(設計情報データベース)[1]に一元管理する。リポジトリに格納する情報を表1に示す。

表1　リポジトリで管理する設計情報（一部）

ドキュメント	管理する設計情報
データフロー図	プロセス名 データフロー名 データストア名 外部名 プロセス仕様
データ構造図	データ構造名 データ要素 データタイプ
モジュール構造図	モジュール名 パラメータ 呼び出しモジュール
モジュール仕様書	モジュール名 パラメータ 呼び出し形式 作業領域 参照共通変数 呼び出しモジュール
プログラム情報	ソースファイル情報 型定義情報 データ定義情報 ニーモニック情報

(6)分野知識データベース

分野特有のデータ項目および処理のパターンを整理して格納したデータベースである。格納されている情報を以下に示す。

1)データ項目名
2)データ項目名の意味情報
3)データ項目名のニーモニック名
4)処理のパターン
5)処理パターンのコード生成手順

3. 特徴的な機能

SoftDAで提供する各工程間を連携する特徴的な機能を以下に示す。図1に連携する機能の関係を示す。

3.1. 日本語要求仕様文からのデータフロー図、データ構造図の生成

従来、設計者が日本語要求仕様文から機能およびデータの関係を抽出してデータフロー図およびデータ構造図を作成していた。しかし、理解および関係抽出に手間がかかり効率的でなかった。

構造化分析を効率化するため、以下の手順で日本語要求仕様文からデータフロー図およびデータ構造図を自動的に生成する[2]。

(1)日本語文から単文および単語間の関係を抽出
(2)この関係に基づきデータフロー図とデータ構造図の雛型を生成
(3)単語の意味に基づきデータフローを補間

日本語要求仕様文に抜けおよび誤りが含まれている場合には、設計者による手直しが必要となる。文章のままではこれらの誤りを検出することは困難であるが、本機能で図式化することにより、仕様の抜けや誤りが見つけ易くなる。

図2に日本語要求仕様文から作成したデータフロー図およびデータ構造図の例を示す。

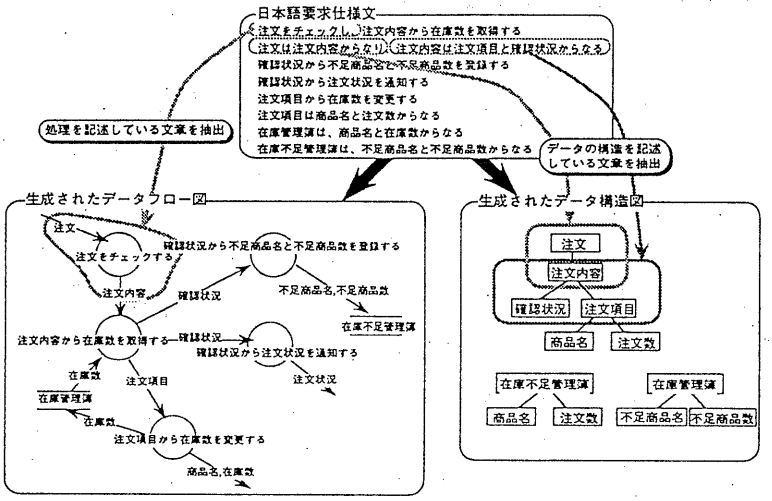


図2　日本語要求仕様文からのデータフロー図、データ構造図の生成例

3.2. データフロー図からのモジュール関連図の生成

データフロー図からモジュール関連図を自動生成する[3,4]。生成するデータを以下に示す。

1)モジュール名
2)パラメータ
3)呼び出し関係

従来のCASEツールで実現されているモジュール関連図生成機能では、人手で作成したプログラム構造と隔たりのある不自然なモジュール関連図を生成する場合があった。本機能ではこの問題点を解決するため、以下に示す生成アルゴリズムを採用している。

(1)データフロー図のプロセス種別を推定

入出力データフローをもとにプロセスを入力、出力、変換の３種類に分類する。

(2)データフロー図をモジュール関連図に変換

プロセスの接続関係に従って、プロセスを分類する。

1)データフローが一方向に流れるプロセス
2)データフローが循環するプロセス

前者はプロセス種別と接続関係の組み合わせに応じた親子関係を持つモジュールの組を生成し、後者は平坦なモジュール関連図を生成する。

図3にデータフロー図からモジュール関連図を作成する例を示す。

3.3. モジュール仕様書の生成

従来は、モジュール関連図を参照しながら設計者が手作業でモジュール仕様書を作成作成していた。このため、モジュール名、パラメータ、呼び出しモジュール等の二重投入による転記ミスなどによりモジュール関連図とモジュール仕様書の不整合が発生していた。

これらの問題点を解決するため、モジュール関連図からモジュール仕様書のインタフェース情報（呼び出し形式、パラメータ情報）および処理手順のスケルトンを生成する。この処理では、モジュール関連図およびデータ構造図の以下の情報を参照する。

1)モジュール呼び出し関係
2)モジュール入出力データ
3)データ構造
4)制御構造

モジュール関連図からモジュール仕様書を作成する例を図4に示す。

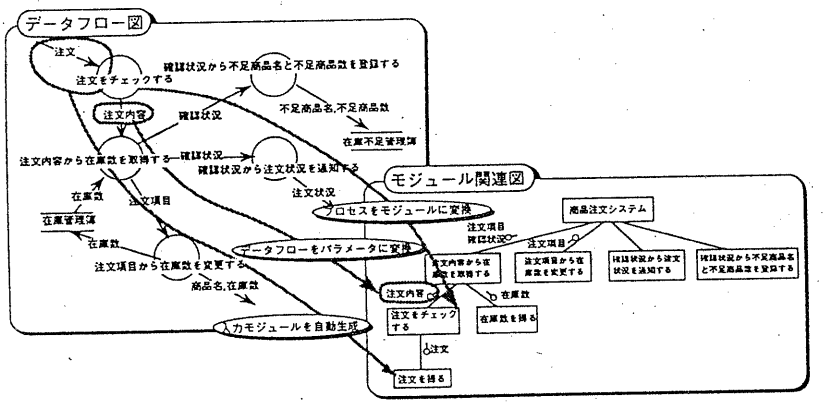


図3　データフロー図からのモジュール関連図の生成例

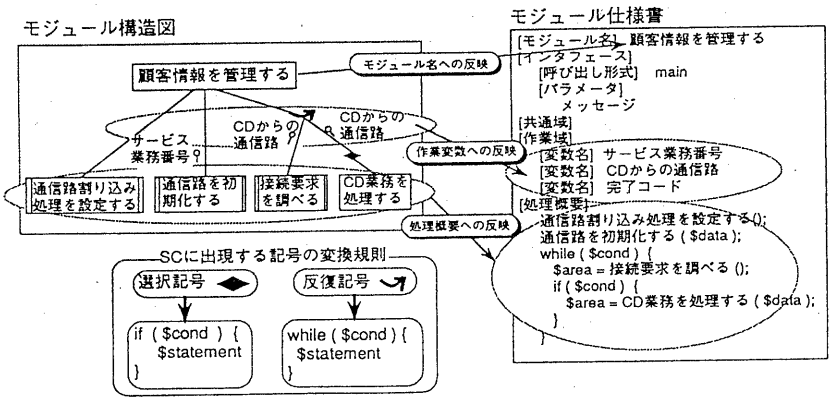


図4　モジュール仕様書の自動生成例

3.4. 詳細処理手順の入力支援

3.4.1. 処理概要の入力支援

共通・汎用的なシステムライブラリおよび対象アプリケーション分野のユーザライブラリを日本語疑似コーディング表現のパターン（テンプレート）で分野知識データベースに登録しておき、再利用する。テンプレートはキーワードと呼ぶ固定部分と、可変部分を表わす穴(ホール)から構成される。詳細設計ではテンプレートを選択し、テンプレートのホールに埋めるデータ項目を決定することにより設計を進める。

詳細設計で使用されるデータ項目のうち上流で作成された項目については、設計情報を一元管理しているリポジトリからデータ項目を抽出してメニュ形式で選択して展開する。これによりモジュールの処理概要と上流の設計情報との一貫性を保証できる。上記の処理で作成されなかったデータ項目はローカルな作業変数である。これらについては参照するモジュールに定義文を生成し、データ項目をリポジトリに追加する。以降はメニュ選択が可能になる。

詳細設計イメージを図5に示す。

テンプレートの種類を以下に示す。

1)制御テンプレート

if,while,for,do,switch,return,break等のプログラム制御構文に関するテンプレート。

2)処理テンプレート

関数、手続き(モジュール呼び出し)のテンプレート。

システムライブラリ、ユーザライブラリ等の利用パターンを日本語疑似コーディング表現のテンプレートとして登録する。

3)代入テンプレート

代入文を表わすテンプレート。

【例】テンプレート

1)制御テンプレート

```
if ( $cond ) {   -- $condは式が展開できるホール
    $statement  -- $statementは文が展開できるホール
}
```

2)処理テンプレート

```
通信路を初期化する( $data );
```

3)代入テンプレート

```
$data = $exp ;
```

3.4.2. データ項目の入力支援

テンプレートのホールに埋めるデータ項目は、上流で作成された設計情報を一元管理しているリポジトリから抽出する。しかし、大規模ソフトウェアではデータ項目が膨大であるため、データ項目一覧表からの選択は容易でない。

このため、テンプレートのホールに埋めるデータ項目の候補を適切に絞り込む必要がある。そこで対象とする分野で標準的に用いられるデータ項目をその意味に基づき整理し、これを属性として定義しておく。データ項目の入力支援ではこの情報をもとにデータ項目を絞り込む。また、属性を付加するオブジェクトを以下に示す。

1)テンプレート
2)ホール
3)データ項目

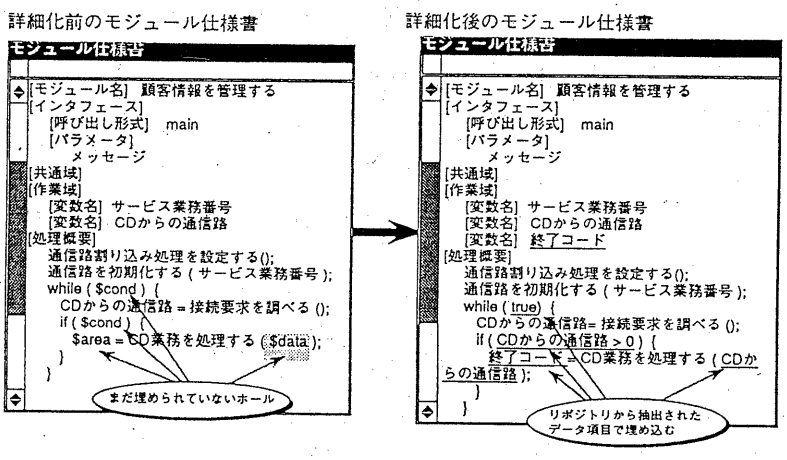


図5　詳細化作業例

候補を絞り込むため、以下の手順で実現する。

1)リポジトリから対象データ項目を抽出
2)テンプレートホールの属性を抽出
3)データの属性を抽出
4)属性が一致するデータ項目のみを候補として採用

絞り込み例を図6に示す。

3.5. 修正支援

修正支援とは、設計ドキュメントのある部分に追加、削除、置換等の修正を加えた時に、他の部分を自動的に修正したり、修正の影響を受ける箇所を解析する機能である。

設計情報を修正した場合、他に及ぼす影響には次の2種類がある。

1)修正作業工程以前の上流工程に及ぼす影響
2)修正作業工程以降の下流工程に及ぼす影響

したがって、修正波及方向として上流工程から下流工程へと下流工程から上流工程への2方向がある。それぞれに対して以下の支援機能を提供する。

(1)下流から上流工程への修正波及

下流工程の修正を上流工程の設計情報に自動的に反映することは難しい。このため、上流工程に影響を及ぼす修正については、修正するオブジェクトの出現箇所を検索し影響箇所を指摘するにとどめる。設計言語エディタでは定義参照箇所に基づく編集機能を提供する。

(2)上流から下流工程への修正波及

上流工程で修正する場合には単に下流工程の設計情報を再度自動生成すればよいように思える。しかし、下流工程では上流工程に影響を及ぼさない修正および上流工程の設計情報と連携しない新たな設計情報の作成がある。このため、上流工程で修正された情報を下流工程に自動的に反映する場合、下流ドキュメント上で施した修正を活かす機能が必要である。

設計情報間の一貫性規則に従い、上流工程の設計情報の修正に対応して下流工程の設計情報を自動的に修正する。自動的に修正された部分にはマークづけを行い、設計言語エディタ上でマークに基づく編集機能を提供する。下流工程で投入された設計情報はそのまま保存する。

下流工程で投入された設計情報は、上流工程で設計したオブジェクトとの対応関係に基づき編集する。すなわち、対応する上流オブジェクトが上流での修正で不変であれば、下流で投入した情報はそのまま保存する。一方、上流オブジェクトが更新されている場合は下流で投入した情報はそのまま保存すると同時にマークづけする。

オブジェクトに付加するマークには以下の2種類がある。

(1)追加マーク
(2)削除マーク

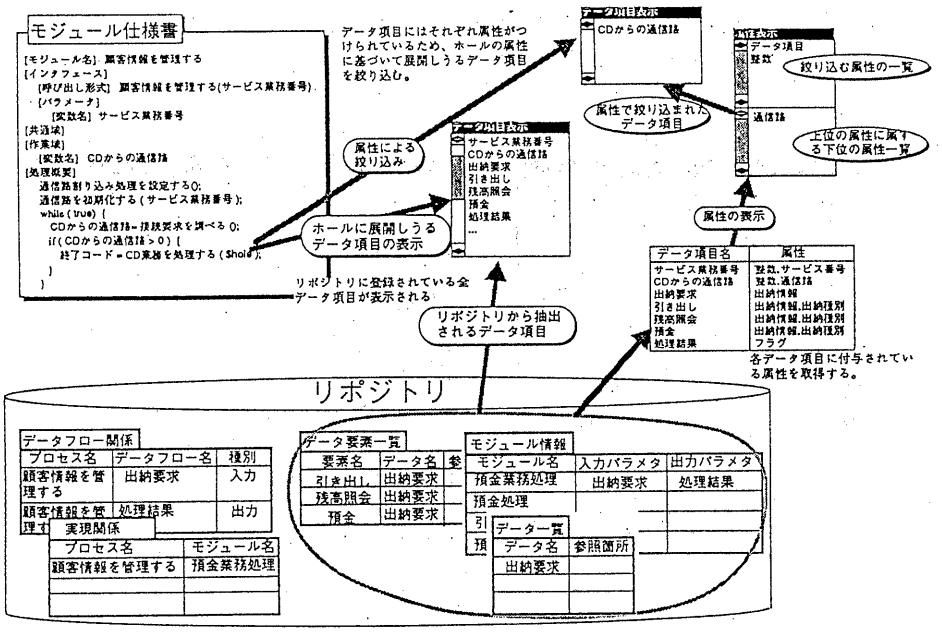


図6　候補語句絞り込み例

このマークに基づく編集機能を以下に示す。

(1)マークの削除

(2)削除マークのつけられたテキストの削除

(3)マークの検索

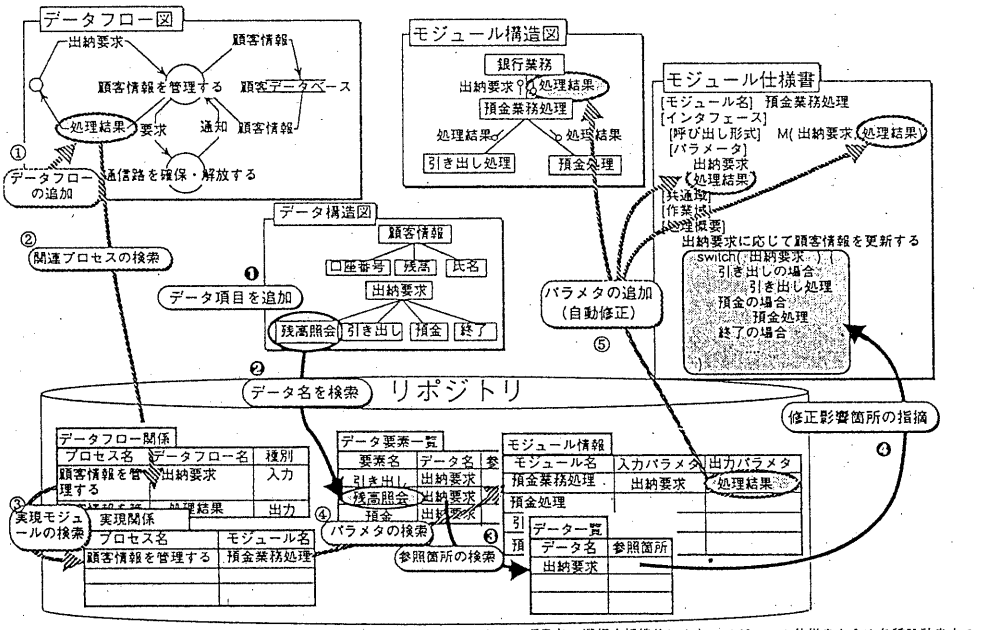
修正支援機能の例として、モジュール関連図上での修正内容をモジュール仕様書へ反映する再生成機能がある。モジュール関連図上で呼び出し関係やパラメータを変更した場合、対応するモジュール仕様書に修正内容を自動的に反映する。また、モジュール仕様書で追加した情報がモジュール構造図上で対応がとれる情報であれば、モジュール仕様書に対してマークづけを行う。

リポジトリを用いた自動修正および影響箇所の指摘例を図7に示す。修正作業例は

1)データフロー図でデータフローを追加

2)データ構造図でデータ項目を追加

である。



データ項目名の選択支援機能により，モジュール仕様書を含め各種設計書中のデータ名の参照は，すべてリポジトリへの参照で実現されている。

図7　リポジトリを用いた自動修正および影響箇所の指摘

3.6. ソースコードの自動生成

日本語疑似コーディングされた処理概要からC言語ソースコードを生成する。コード生成手順を以下に示す。

(1)モジュール仕様書から関数定義のスケルトンを生成

(2)変数宣言を生成

・データ構造図から型を抽出

・型名、データ項目名をニーモニック化

(3)処理概要をコード化

・テンプレートをソースコードに変換

・データ項目をニーモニック化

型名、データ項目およびモジュール名等の日本語名とニーモニックの対応表は分野知識データベースに格納されている。日本語疑似コーディングされた処理概要からコードを生成する例を図8に示す。

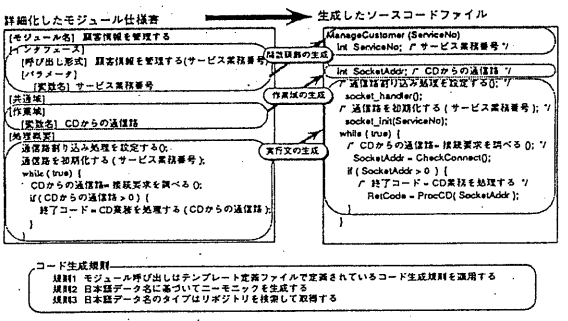


図8 ソースコードの自動生成

3.7. 疑似コードレベルでのデバッグ

疑似コーディングされた処理概要と生成コードとの対応をとることにより、デバッグをモジュール仕様書上で行える。このため、ソースコードを意識することなくデバッグできる。Cプリプロセサの行制御マクロを使用して、生成したコードに対応する処理概要の疑似コード上の記述位置を一対一に対応づける。疑似コードに対して操作できる主な機能を以下に示す。

(1)トラップ文の指定
(2)データ項目の値の印刷およびトレース
(3)ステップ実行

疑似コードレベルでのデバッグ作業イメージを図9に示す。

4. 効果

統合化機能の実現により以下の効果がある。

(1)上流工程の設計ドキュメントから下流工程の設計ドキュメントを自動的に生成
  1)インタフェースおよび処理概要の二重投入が不要
  2)コーディング作業が不要
(2)上流上程で投入した設計情報を活用した開発支援
  1)キーボードレスな開発
  2)ソースコードレスな開発

5. 評価

5.1. 処理概要の入力支援

ライブラリモジュールを分野毎に階層的に分類しているため、使用するモジュールの検索および選択がメニュのみで行える。従来はキーボード入力で行っていたがメニュによる選択で入力効率が向上した。また、キーボード入力に伴うキー入力ミスが皆無である。

5.2. 候補提示および絞り込み

ホールに展開しうるデータ項目を型によって絞り込むことにより、候補となるデータ項目を絞り込めるのでデータ項目の選択が容易となった。

あるプログラム(49関数、5.3Ks)では、データ項目数が338個、型数が90であった。したがって、絞り込み機能により選択対象が平均90分の1と大幅に減少した。なお、1つの型当たりのデータ項目数は1から11個(平均3.75個)であった。

6. おわりに

本稿ではCASEツールの統合化機能について述べた。本稿で述べた自動化機能による生産性向上効果については今後評価を行う。

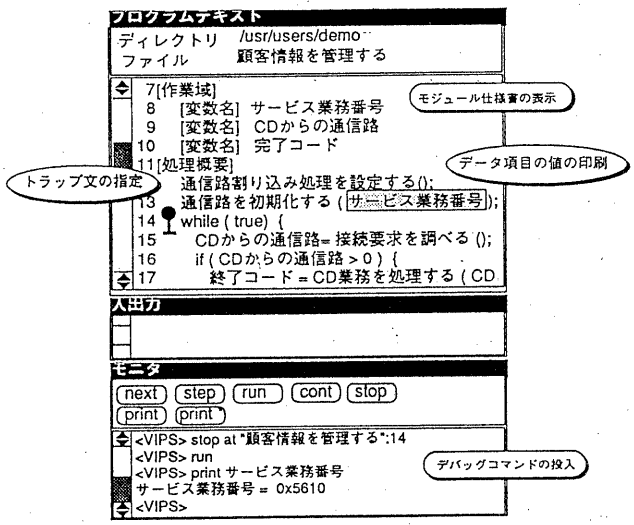


図9 疑似コーディングレベルでのデバッグ作業例

参考文献

[1]岡,山本,磯田：設計情報リポジトリにおける一貫性管理機能,情報処理学会ソフトウェア工学研究会(1991)80-6

(2)三浦,山本：日本語文からのデータフロー図作成支援機能の提案,情報処理学会第43回全国大会5-387(1991)

(3)西永,山本,磯田：モジュール構造設計支援機能の提案,情報処理学会ソフトウェア工学研究会(1991)81-2

(4)Yamamoto,Isoda:ATransformation Algorithm from DataFlow Diagrams into Structure Charts, Joint Conference on Software Engineering(1992)

(5)黒木,斎,高橋,鶴岡,山本：ソフトウェア分析設計支援ツールの統合化,情報処理学会第43回全国大会5-421(1991)