ソフトウエア工学 34−13
（１９８４．２．１０）

日本語をベースにした仕様記述言語ＮＢＳＧにおける手続記述について

杉尾俊之　武内惇　椎野努
（沖電気工業株式会社　システム本部）

１．はじめに

近年、計算機の日本語処理技術の進歩により、わかりやすいプログラムを作るために日本語を使用する試みが進められている。しかし、それらの多くは既存の高水準プログラム言語の制御構造をそのまま踏襲しており、日本語を使用しているものの、わかりやすいプログラムを設計するという点では、今後更に改善の余地がある。特に、ループの制御変数などハードウェア上での処理実現のための物理的表現（処理の流れを制御するためだけに使用される表現）が処理手続きの記述の中に混在するためにプログラムが理解しにくいものとなっている。また、分岐条件の設定方法に一貫性がないことが、条件の定義漏れ、二重定義などの矛盾を生じ、デバッグ、保守時の作業量を増す原因となっている。

我々は、設計者の処理手続きの思考過程に適合した制御構造の表現法を提供し、日本語によるわかりやすいプログラムモジュールの設計を実現するという観点に立ち、日本語による手続き記述言語（ＮＢＳＧ／ＰＤ）を開発した。ＮＢＳＧ／ＰＤは日本語をベースにした仕様記述言語ＮＢＳＧ[1][2][4]と結合することにより、機能仕様（要求仕様）定義から処理仕様定義までの開発支援を可能とする。

本稿では、ＮＢＳＧ／ＰＤの設計方針を示し、その構文および設計を支援する機能について述べる。

２．ＮＢＳＧのプロフィール

従来のユーザ要求仕様は、非制限的な（構文が規定されてない）自然言語で記述され、その書式が明確に示されていなかったため、仕様記述のあいまいさ・冗長・矛盾・脱漏等が避けられなかった。

これらの問題を解決するためには、仕様の記述方法にある程度の制限を設け、これにより計算機によるサポートを可能とする必要がある。ＮＢＳＧはそのような要求を満たすべく開発され、設計ガイダンスおよび静的解析機能などの設計支援機能を備えている。

現在、ＮＢＳＧ支援系は、要求される要素機能の抽出、および、データ類の図表表現[5]、そして今回報告する要素機能の処理手続きの表現法（ＮＢＳＧ／ＰＤ）へとその適用範囲を拡張しつつありソフトウェア開発作業全般を対象とした総合ソフトウェア作成支援システムとしての実現をめざしている。

３．ＮＢＳＧ／ＰＤの設計方針

ＮＢＳＧ／ＰＤを設計するにあたって、

(1)設計しやすいプログラム言語の実現
(2)読みやすく理解容易なプログラム言語の実現
(3)設計および保守を支援する機能の実現

の３点に着目し、その設計方針とした。

(1)設計しやすいプログラム言語の実現

設計者にとって設計しやすいプログラム言語を実現するために、以下のことが重要である。

①処理手続き表現の思考過程に適合した論理的な記述

手続きの制御構造として、シーケンス・分岐・繰り返しの３つの基本構造を採用し、それぞれ１入口１出口のブロック構造で表現する。

特に、分岐条件の設定方法を統一し、条件の漏れを除去する。

②入出力データの自動生成

要素機能間のインタフェースは、既にＮＢＳＧ図表表現で定義されており、ＮＢＳＧ／ＰＤ記述時には、その情報を利用して各要素機能体ごとの入出力データの宣言文が自動生成される。これにより、

・設計ガイダンスを行なうことができる。
・矛盾した定義を防ぐことができる。

③語いを限定した習得容易な構文

言語の習得容易性は、記述されたプログラムの質に影響を与える。ＮＢＳＧ／ＰＤでは、標準語いの数を制限し、できるだけ平易で自然な日本語構文を採用することにより、言語そのものの習得を容易にして設計者の負担を

軽減する。

④制御変数の隠蔽を図るための方策

配列の添字、ループの制御などに使用される制御変数は、元来、計算機のハードウェアに依存する物理的表現であり処理手続きの論理的な表現とは区別して扱うべきものである。ＮＢＳＧ／ＰＤでは、データセットのブロック転送などのマクロ表現によって制御変数の隠蔽を図り、論理的な処理の流れを表現するためには不必要な表現（物理的表現）をできるだけ除去する。

(2)読みやすく理解容易なプログラム言語の実現

読みやすく理解容易なプログラム言語の実現については、次の点に着目した。

①平明な日本語による表現

自然言語風の擬似コードによる処理手続きの表現は、その語いの多様性からあいまいな表現となることが多い。また、日本語を使用したプログラム言語でも、従来の英語ベースの高水準プログラム言語をそのまま日本語に直しただけのものでは、日本語の特質を十分発揮できず読みにくいものとなる。ＮＢＳＧ／ＰＤでは、処理手続きを表現するのに語いを制限し、日本語の文法をベースとした自然な文章でプログラムを表現することで理解容易性を高めようとしている。

これにより、処理の斜め読みを可能とするプログラム言語の記述が可能となる。

②制御変数の扱い

処理手続きの論理レベル表現の中では、制御変数を安易に使用すべきでないが、どうしても必要な場合は、制御変数を英文字列とし、日本語で記述された処理手続き論理の表現と区別する。

③簡明な制御構造

構造化されたブロック構造は人間の思考形態に適合し、設計しやすいのと同時に理解しやすいものとなる。このことは、誰が読んでも同様に解釈されるあいまいさのないプログラム言語を実現する。

(3)設計および保守を支援する機能

ＮＢＳＧで定義された機能に関する様々な情報を利用できること、ＮＢＳＧ／ＰＤが支援機能の実現を意識した構文特性を有することにより、次にしめすような支援機能が実現できる。

①エディタ機能
②静的解析機能
③動的解析機能

4．ＮＢＳＧにおける手続表現（ＮＢＳＧ／ＰＤ）の位置づけ

ＮＢＳＧでは、機能仕様をトップダウンに定義していく。その際、機能およびそれに関係するデータ類は階層的に詳細化される。詳細化が繰り返された結果、出力データを生成するための入力データの変換論理が明らかである様な要素機能が抽出される。この要素機能のデータ変換論理（処理手続き）を記述するのがＮＢＳＧ／ＰＤである。（図１）

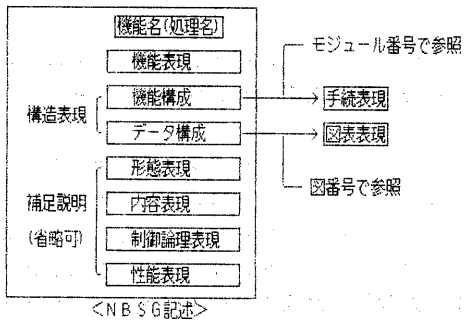


図1：手続表現（NBSG/PD）の位置づけ

ＮＢＳＧ／ＰＤの記述は、ＮＢＳＧで定義された要素機能とモジュール番号によって対応づけられる。

また、ＮＢＳＧ／ＰＤで記述されるモジュールの入出力データはＮＢＳＧ図表表現に定義されている情報を利用する。

5．ＮＢＳ.Ｇ／ＰＤの構文

(1)文書構成

ＮＢＳＧ機能構成とモジュール番号で対応づけられたＮＢＳＧ／ＰＤ（手続表現）は以下の表現形式を用いて記述される。（図２の(a)）

①モジュール名

ＮＢＳＧの機能構成表現で抽出された要素機能（機能体構造木の最下位に位置する機能体）の名称をモジュール番号とともに記述する。

②入力データ、出力データ

ＮＢＳＧ図表表現で定義された情報を利用して自動生成されるので記述する必要はない。

③内部データ

モジュール内で作業用に使用する内部データに関する属性、初期値の設定を行なう。

④関数

ＮＢＳＧ／ＰＤでは処理手続きの共用化を図るために関数の定義を認める。モジュールで使用する関数名をここに定義しておく。

⑤手続き記述

処理手続きをＮＢＳＧ／ＰＤの構文に従って記述する。

また、関数の定義は図２の（ｂ）に示す形式で行なう。

①関数名

定義する関数の名称を記述する。

②入出力パラメータ

関数に渡されるパラメータの属性および関数から呼出し側へ返されるパラメータの属性を定義する。

③内部データ、関数、手続き記述についてはモジュールの定義と同様に行なう。

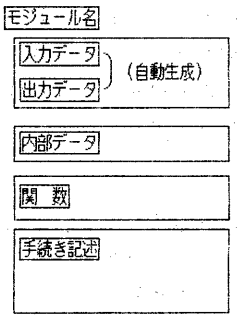


図2（a）：モジュールの文書構成

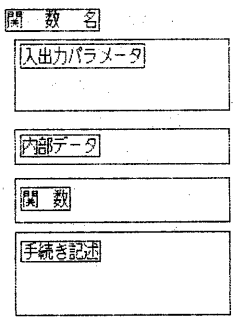


図2（b）：関数の文書構成

(2)構文の概要

ＮＢＳＧ／ＰＤは、シーケンス・分岐・繰り返しの３つの表現からなる。[3]

①シーケンス表現

シーケンス表現は、処理のシーケンシャルな手続きを記述するためのものであり、次の３種類の文で構成される。

・入出力文：データの転送を記述する。

・実行文：関数の実行を記述する。

・算術演算文：算術演算を記述する。

シーケンス表現は単文を用い、句点で区切られる。

②分岐表現（図３の（ａ））

分岐表現は、条件によって処理の流れを変えるためのもので、条件の設定漏れを防止するために必ず「その他のとき」の条件に対する記述を行なわなければならない。

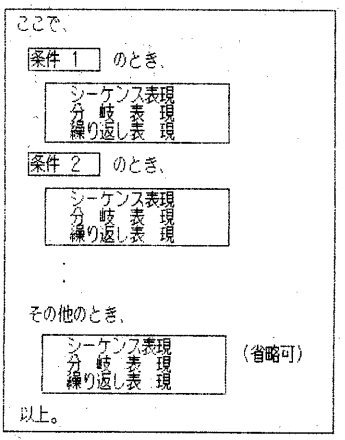


図3（a）：分岐表現の構文

③繰り返し表現（図３の（ｂ））

繰り返し表現は、繰り返し条件の設定方法によって２つの場合がある。

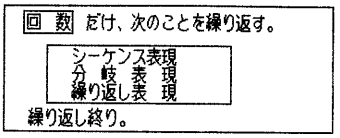
・回数指定の繰り返し：

繰り返しの回数を一意的に決めることができる場合に回数をカウントする制御変数を使用することなく繰り返しを記述する表現である。

・条件指定の繰り返し：

比較式などで繰り返しの条件を指定し、その条件の真偽によって繰り返しの継続を制御する場合に使用する。

回数指定の場合



条件指定の場合

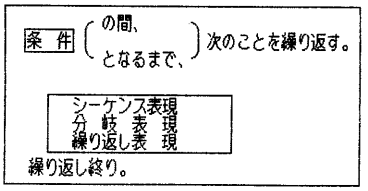


図3（b）：繰り返し表現の構文

また、ＮＢＳＧ／ＰＤで使用される変数名・関数名は日本語（かなおよび漢字）を用いて記述する。ただし、制御変数は英文字列で表わし、これによって、処理の流れを制御するための物理的表現を区別して認識することを容易にする。表１にＮＢＳＧ／ＰＤで使用される変数の種類を示す。

表1：ＮＢＳＧ／ＰＤに使用される変数一覧

	変数	説明	通用範囲
モジュール	入出力変数	図表表現より自動生成される。	モジュール間（大域的）
	内部変数	モジュール内で作業用に使用する変数。	モジュール内（局所的）
関数	入出力パラメータ	関数呼び出しのフォーマルパラメータとして定義する。	関数内（局所的）
	関数値	関数の処理結果の値が入る。	関数内（局所的）
	リターンステータス	関数に付随する属性の表示データ。	関数内（局所的）

(3)データセットのブロック転送

ＮＢＳＧ／ＰＤの設計方針の一つに示したように、処理手続きの表現中に物理的表現を混在させることはプログラムを理解しにくくする。特に、制御変数に関する表現を隠蔽することは、人間の思考形態に適合した処理手続きの設計には不可欠である。ＮＢＳＧ／ＰＤでは、配列の添字のアクセス法に次に示す規則を導入し、単純なデータセットのブロック転送に関しては制御変数を不要とした。

①配列の添字が「＆」で始まる場合は、それがアクセス時の初期値であることを示す。

②多次元配列において「＆」が付与された添字が複数個存在する場合は、最も右に位置する添字から順にアクセスされる。

データセットの転送例を図４に示す。

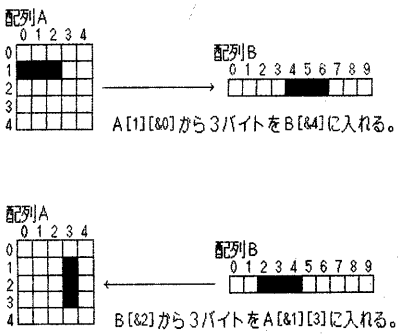


図4：データセットの転送例

(4)関数の呼び出し／実行

表１に示したように、関数を呼び出した際に受け渡されるデータには、入出力パラメータの他に関数値とリターンステータスがある。

①関数値

その関数の処理結果の値が入り、呼び出し側では関数名によって参照できる。また、呼び出された関数内での関数値の設定は、その関数名を左辺にした算術演算文で行なう。（図５（ａ））

②リターンステータス

関数の実行に付随する属性を表示するためのデータ（リターンコード）で、関数名を〈と〉で囲むことによって参照可能である。関数内でリターンステータスを設定するためには、キーワード（'STATUS'）にステータスの値を代入すればよい。（図５（ｂ））

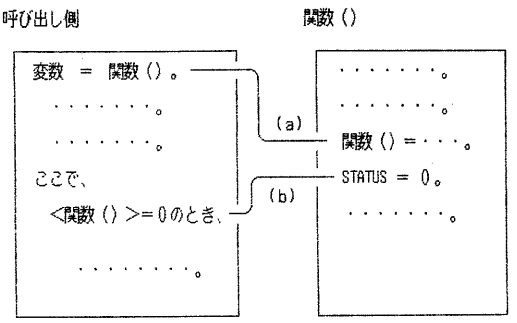


図5：関数値、リターンステータス

6．ＮＢＳＧ／ＰＤ設計支援系に要求される機能

ＮＢＳＧ／ＰＤは、ＮＢＳＧ処理系の１部分として実現される予定であるが、その処理系（設計支援系）を設計するに際して要求される機能を抽出した。

(1)エディタ

ＮＢＳＧ／ＰＤで設計作業を行なう際に必要となるのがエディット機能である。その要件は、設計者の入力作業量を削減し、修正や編集作業を容易にすることである。ＮＢＳＧ／ＰＤの特長を十分生かすために、シンタックスオリエンテッドなエディタとして実現する。機能として必要なものを次に示す。

①構文に促して出力されたガイダンスで指示された部分のみを記述（入力）する方式。

②入力に必要な各種情報（入出力データ、関数の使用状況など）を表示する機能。

③変更、修正された記述を管理する版管理機能。

(2)デバッグ機能

記述されたＮＢＳＧ／ＰＤの論理的な正しさを検証することを支援する機能で、静的解析機能と動的解析機能に分けられる。

①静的解析機能[6]

実行テストの前の段階で、ＮＢＳＧ／ＰＤで記述されたソースコードの内容をチェックすること（ケアレスミス、誤りやすい所の指摘など）は、作業効率、プログラム品質を向上させる上で有効である。

ＮＢＳＧ／ＰＤでは一般的な静的解析の他に特徴的なものとして、次にしめすチェックを行なう。

・出力変数生成論理の抽出：

ＮＢＳＧ／ＰＤ中のある変数に着目し、その変数に関係する表現のみを抽出することは、設計レビューを容易にすると考えられる。

・関数の入出力パラメータのインタフェースチェック：

関数の呼び出し側で指定したパラメータと実際に関数の方で定義されているパラメータの属性を比較し矛盾を検出する。

・処理手続きの実行ルートの抽出：

漏れ、重複のないテストケースの設定を支援するため、処理手続きの実行ルートの抽出を行なう。

・分岐条件「その他のとき」の条件解析：

例外となる分岐条件「その他のとき」の内容を具体的に示し、分岐条件の誤りの検出を容易にする。

②動的解析機能

実行テスト段階で誤りが確認された場合、実行状態を的確に把握するためのデータを収集することは、原因を早急に発見するために有効である。ＮＢＳＧ／ＰＤでは、その特長を利用し、次に示す動的解析を行なう。

・アサーション文の自動生成：

ＮＢＳＧの図表表現で定義された入出力変数の値域、変数間の関係情報をもとにデバッグ用のアサーション文を自動生成する。

・モジュール／関数の実行レベルでのトレース：

ＮＢＳＧ／ＰＤは、ＮＢＳＧの機能構成で抽出された要素機能（モジュール）の処理手続きを記述するが、それぞれのモジュールの実行順序、同期関係については、ＮＢＳＧの機能表現の制御・状態によって定義されている。したがって、設計しようとしているシステム全体の動作状況を把握するためには、それらの情報を人間がわかりやすい形で表現する必要がある。

7．おわりに

ＮＢＳＧ／ＰＤを使用することにより、次のような効果が期待できる。

(1)処理手続き表現の思考過程に適合した論理的な制御構造を採用したことにより、設計しやすいプログラム言語となっている。

(2)語いを限定した平易な日本語で表現するので、習得が容易であり、設計者の負担を軽減できる。また、記述されたプログラムの理解性も高まり、いわゆる斜め読み可能なプログラムを実現できる。

(3)制御変数の扱いや分岐表現の設定方法などの文章構造が統制されているので、記述方法が統一され、誰が読んでも同様に解釈されるプログラムを記述できる。

(4)設計支援機能によりＮＢＳＧで既に定義されている様々な情報を利用でき、入出力データの自動生成、記述内容の静的解析や動的解析など、計算機を用いた効率的な設計が可能となる。

ＮＢＳＧは、今後、要求仕様定義から処理仕様定義までの開発環境を支援するシステムとしての実現をめざしている。

謝辞　日頃、御指導いただく九州大学工学部
吉田 将 教授、ならびに 弊社システム本部
中江 康史 企画室長に感謝します。

参考文献

[1] 椎野 他：「日本語をベースにした仕様記述言語への一アプローチ」，情処第24回全国大会，1982.3
[2] 椎野 他：「日本語をベースにした仕様記述言語ＮＢＳＧ」，情処第26回全国大会，1983.3
[3] 椎野 他：「日本語をベースにした仕様記述言語ＮＢＳＧにおける手順記述」，情処第26回全国大会，1983.3
[4] 椎野 他：「日本語をベースにした仕様記述言語ＮＢＳＧ」，情処学会ソフトウェア工学研究会資料(30-2)，1983.6
[5] 椎野 他：「日本語をベースにした仕様記述言語ＮＢＳＧの図表入力について」，情処第27回全国大会，1983.10
[6] 椎野 他：「日本語をベースにした仕様記述言語ＮＢＳＧの手続記述の静的解析機能について」，情処第27回全国大会，1983.10